GAP (GROUP, ALGORITHM AND PROGRAMMING) İLE CEBİR VE SAYILAR TEORİSİ

Alper ODABAŞ

Dumlupınar Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Matematik Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman : Doç. Dr. Murat ALP

Temmuz – 2004

GAP
(GROUP, ALGORITHM AND PROGRAMMING)
İLE CEBİR VE SAYILAR TEORİSİ

Alper ODABAŞ

Yüksek Lisans Tezi

Matematik Anabilim Dalı

Temmuz - 2004

KABUL VE ONAY SAYFASI

Alper ODABAŞ'ın YÜKSEK LİSANS tezi olarak hazırladığı GAP (GROUP, ALGORITHM AND PROGRAMMING) İLE CEBİR VE SAYILAR TEORİSİ başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir

16/7/2004

Üye : Prof. Dr. Zekeriya ARVASİ

Üye : Doç. Dr. Murat ALP

Üye : Doç. Dr. Mahmut KOÇAK

Üye : Yrd. Doç. Dr. İsmail EKİNCİOĞLU

Fen Bilimleri Enstitüsün Yönetim Kurulu'nun 26/07/2004 gün ve ....12..... sayılı kararıyla onaylanmıştır.

Prof. Dr. M. Sabri ÖZYURT

Fen Bilimleri Enstitüsü Müdürü

GAP (GROUP, ALGORITHM AND PROGRAMMING) İLE CEBİR VE SAYILAR TEORİSİ

Alper ODABAŞ
Matematik, Yüksek Lisans Tezi, 2004
Tez Danışmanı: Doç. Dr. Murat ALP

ÖZET

Bu tez on sekiz bölümden oluşmaktadır. Birinci bölüm giriş kısmını oluşturmaktadır. İkinci bölüm GAP (Group, Algorithm and Programming) programının elde edilmesini, internetten indirilmesini ve Windows işletim sistemlerinde bilgisayara yüklenmesini içermektedir. Üçüncü bölüm GAP programlama dilinde kullanılan program komutlarının kısa bir sunumunu içermektedir. GAP programındaki kullanılan fonksiyonların kullanılması örnekleriyle birlikte dördüncü bölümden sunulmuştur. Bundan sonraki bölümler ise Cebir ve Sayılar Teorisi konularına ayrılmış olup bu konuların GAP programı aracılığı ile incelenme örnekleri de bölümler içerisinde verilmiştir.

Bölüm 5, kümeler ve kümelerle ilgili örnekleri içermektedir. Bölüm 6, fonksiyonlar ve fonksiyon kavramıyla ilintili olan bilgilerin GAP programı aracılığı ile sunulmasını içermektedir. Bölüm 7, tamsayılar, OBEB-OKEK konularını sunmaktadır. Bölüm 8, modüler aritmetik kavramının GAP uygulamasının örneğiyle birlikte sunmaktadır. Bölüm 9, grup teori kavramını ve bu kavramın GAP programı kullanılarak incelenmesini kapsamaktadır. Bölüm 10, alt gruplar konusunu ve konu ile ilgili örnekleri içermektedir. Bölüm 11, normal alt gruplar kavramının GAP programı aracılı ile incelenmesini ve GAP programları örneklerini incelenmektedir. Bölüm 12, alt gruplar konusunu ve konu ile ilgili örnekleri içermektedir. Bölüm 13, simetrik grup ve simetrik gruplarla ilgili örnekleri sunmaktadır. Bölüm 14, Sylow teoremlerini içermektedir. Bölüm 15, halkalar, alt halkalar ve idealler kavramlarını GAP programları örnekleri ile birlikte sunmaktadır. Bölüm 16, bölüm tamlık bölgesi, cisimler ve alt cisimler kavramlarının GAP programı kullanılarak incelenmesini içermektedir. Bölüm 17, bölüm halka homomorfizmleri ve GAP programı örneklerini sunmaktadır. Bölüm 18, grup latislerini içermektedir.

Anahtar Kelimeler : Alt Cisim, Alt Grup, Alt Halka, GAP, Grup, Grup Homomorfizmi, Grup Latisi, Modüler Aritmetik, Normal Alt Grup, Simetrik Grup, Sylow Teoremi, Tamlık Bölgesi, Tamsayılar.

ABSTRACT ALGEBRA AND NUMBERS THEORY WITH GAP (GROUP, ALGORITHM AND PROGRAMMING)

Alper ODABAŞ
Mathematics, MSc Thesis, 2004
Thesis Adviser: Assoc. Prof. Dr. Murat ALP

SUMMARY

This thesis consists of eighteen chapters. In the first chapter, the introduction was given. Getting GAP (Group, Algorithm and Programming) program, downloading it from internet and installing it to the Windows operation systems were placed in the second chapter. In the third chapter, a short presentation of GAP program commands were given. Using the GAP functions was presented in the fourth chapter with their examples. The other chapters were set apart for the subjects of algebra and examples of each subject.

Bölüm 5, kümeler ve kümelerle ilgili örnekleri içermektedir. Bölüm 6, fonksiyonlar ve fonksiyon kavramıyla ilintili olan bilgilerin GAP programı aracılığı ile sunulmasını içermektedir. In chapter 5, sets and examples were explained. Functions and presentation of informations about functions through GAP program were located in chapter 6. In chapter 7, integers, greatest common divisor , least common multiple were given. Modular arithmetic and its GAP application were presented in chapter 8. Group theory, and investigation of this concept by using GAP program were presented in chapter 9. In chapter 10, subgroups and their examples were explained. Normal subgroups and investigation of this subject by using GAP were located in chapter 11. Subgroups and their examples were given in chapter 12. In chapter 13, symmetric groups and examples concerned with them were presented. In chapter 14, Sylow theorems were explained. Rings, subrings, ideals and their examples were located in chapter 15. Integral domain, fields, subfields and investigation of these consepts by using GAP program were given in chapter 16. Division ring homomorphisms and their GAP program examples were explained in chapter 17. Finally, in chapter 18, group lattices were presented.

Keywords : GAP, Group, Group Homomorphism, Group Lattice, Integers, Integer Domain, Modular Arithmetic, Normal Subgroup, Subgroup, Subfield, Subring, Sylow Theorem, Symmetric Group.

TEŞEKKÜR

Bu çalışmanın her aşamasında beni destekleyen, hastalığında dahi büyük bir özveriyle beni çalıştıran danışman hocam sayın Doç. Dr. Murat ALP'e, çalışmalarımda bana yardımcı olan Arş. Gör. Adem Cengiz ÇEVİKEL, Arş. Gör. Ali AYTEKİN, Arş. Gör. Ali ŞAHİN, Arş. Gör. Halis BİLGİL'e ve desteğini her zaman hissettiren Serdar YÖNLÜ'ye teşekkürü bir borç bilirim.

İÇİNDEKİLER

İÇİNDEKİLER (devam)

İÇİNDEKİLER (devam)

İÇİNDEKİLER (devam)

GAP FONKSİYONLARI İÇİNDEKİLER

Sayfa

GAP FONKSİYONLARI İÇİNDEKİLER (devam)

Sayfa

GAP FONKSİYONLARI İÇİNDEKİLER (devam)

Sayfa

GAP FONKSİYONLARI İÇİNDEKİLER (devam)

Sayfa

GAP FONKSİYONLARI İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
Δ	Simetrik fark.
$\varnothing$	Boş küme.
$\cup$	Birleşim.
$\cap$	Kesişim.
0_H	H nın sıfır elemanı.
1_H	H nın birim elemanı.
$[a,b]$	a ve b nin komütatörü.
e	Birim eleman.
Euler φ	Euler Phi fonksiyonu.
$\lvert G:H\rvert$	H nın G deki indeksi.
$\lvert G\rvert$	G nin mertebesi.
$Kar(H)$	H nın karakteristiği.
$M(G)$	G nin merkezi.
$N(H)$	H nın normalleştiricisi.
$p(n)$	n nin tüm ayrışımları sayısı.
$T(H)$	H nın tersinerleri kümesi.

Kısaltmalar	Açıklama
GAP	Group, Algorithm and Programming
LDFM	Lehrstuhl D für Mathematik
OBEB	En Büyük Ortak Bölen
OKEK	En Küçük Ortak Kat

1 GİRİŞ

Bu tez günümüzde kullanılan Cebir ve Sayılar Teorisi dersinin bilgisayar aracılığı ile incelenebilmesi için gerekli olan, bilgisayar programı GAP (Group, Algorithm and Programming) hakkında bilgiler ve bu programın fonksiyonlarının kullanımıyla ilgili detaylı açıklamaları sunmaktadır. Tez toplam 18 bölümden oluşmakta olup ilk dört bölüm GAP programının elde edilmesini, yüklenmesini, GAP ortak paket programlarını ve GAP programının temel kullanım komutlarını içermektedir. Diğer bölümler ise Cebir ve Sayılar Teorisinde kullanılan temel konuların GAP programı komutlarıyla incelenmesini içermektedir.

1.1 Neden Cebir ve Sayılar Teorisi

Bu tezin oluşturulmasında Cebir ve Sayılar Teorisi konusunun bilgisayar ortamında incelenerek okuyucuya bilgisayar aracılığı ile bu teorik konuların uygulamasının nasıl olacağı açık bir dille anlatılmanın yanı sıra bu konulardaki teorik örneklerin bilgisayar ile hesaplanması da yer almaktadır.

Cebirin soyut konularının somut bir uygulamasını vermek için Cebir ve Sayılar Teorisi konularının bilgisayara uygulanması seçilmiştir. Seçilen konular ve uygulama örnekleri bölümler içerisinde sunulmuştur. GAP programının seçilmesi ise Neden GAP başlığı altında sunulmuştur.

1.2 Neden GAP

GAP (Group, Algorithm and Programming) cebirin bilgisayar aracılığı ile hesaplanması için geliştirilen bir bilgisayar programıdır. GAP, Lehrstuhl D für Mathematik (LDFM) RWTH Aachen, Almanya'da 1986-1997 yılları arasında geliştirilen bir programdır. LDFM nin başkanlığını yapan J. Neubüser'in emekli olması sonucu GAP, St Andrews Üniversitesi Matematik ve Bilgisayar Bölümü tarafından yönetilmeye başlamıştır. Şu anda GAP programının 1997 den 1999 yılına kadar geliştirilmiş olan versiyonlarından sonra 1999 dan sonra kullanılan ve gelişimini ortak paket programlarının çekirdek denilen GAP programına eklenmesiyle GAP 4.3 versiyonu da yer almaktadır. GAP programı internet ortamından ücretsiz olarak alınıp bilgisayara yüklenebilir bir sisteme sahiptir. GAP programı Unix, Windows ve Macintosh sistemlerinde çalışabilir ve minimum 32 MB lık boşluk ister. Tüm ortak paketlerin yüklenip çalıştırılması istenildiğinde ise 320 MB lık yer kaplar. GAP programını çalıştırabilmek için minimum 20 MB RAM çok gelişmiş hesaplamalar içinde 128 MB RAM yeterlidir. GAP

programı bu türdeki diğer programlama dillerine göre daha düşük konfigürasyonlu bilgisayarlarda ve tüm işletim sistemlerinde çalışır.

GAP programı kullanıcıları kendi içlerinde bir çalışma grubu oluşturarak E-mail aracılığı ile sorular sorulmakta ve üyelerden sorular hakkında çeşitli cevaplar rahatlıkla alınabilmektedir. Bu nedenle cebir konularının bilgisayara uygulanması esnasında rahat bir tartışma ortamı bulunabilmektedir.

Ayrıca GAP programı gruplar teorisi açısından detaylı bilgilerle donatılmış olup Grup Teoride GAP programından başka bu denli donanımlı başka bir program bulunmamaktadır. GAP programlama dili oldukça kolaydır. Ayrıca GAP programında uzantıları .g , .gi olan dosyalara da girilerek matematiksel bir ifadenin bilgisayara uygulanması mantığı da incelenebilir olmasından dolayı GAP programı tercih edilmiştir.

GAP programında program yazımındaki kodlama harf harf yapıldığından kodlama işi uzunca bir zaman almaktadır. Bu da GAP programının dezavantajıdır.

Günümüzde varolan bilgilerin bilgisayar ortamına aktarılması nedeniyle, Cebir ve Sayılar Teorisi konularının da bilgisayar ortamında dahil edilebilmesi için bu konuların GAP ortamında ki incelemeleri bu tezde yapılmıştır.

• Bölüm 5, Kümeler konusu ve kümeler konusunu bilgisayar aracılığı ile hesaplayacak olan GAP programı fonksiyonları örnekleriyle birlikte bu bölümde incelenmiştir. Kümelerle ilgili olan matematiksel konular çeşitli kitaplardan alınmış olup bu alınan tanım, teorem ve örneklerde GAP programı aracılığı ile yeniden oluşturulmuştur. Cebir ve Sayılar Teorisi konularının konu düzeni [7] den alınmış olup bu düzene göre tez şekillendirilmiştir.

• Bölüm 6, Fonksiyonlar kavramı Kümeler konusuna ayrılmış olan bu bölümde ınksiyonların bilgisayar tanımlamaları yapılırken grup yapılarından faydalanılmıştır. Çünkü ınksiyonların tanım ve değer kümeleri tanımlamaları GAP programında gruplarda ıpıldığından bu bölümde gruplar tanıtılmadan fonksiyonların var olan özelliklerini ınıyabilmek için gruplar kullanılmıştır. Tanımlanan fonksiyonların birebir fonksiyon, örten ınksiyon, ters fonksiyon, görüntü ve bileşke fonksiyon özellikleri GAP programının `sInjective()`, `IsSurjective()`, `InverseGeneralMapping()`, `Image()` ve `ompositionMapping()` fonksiyonlarıyla test edilmiştir.

• Bölüm 7, Tamsayılar konu başlığı altında, OBEB-OKEK kavramlarını içermektedir. Bu bölümde Euclid algoritması, en büyük ortak bölen, en küçük ortak bölen, aralarında asal ve Euler φ fonksiyonu konuları bazı GAP programı örnekleri ile birlikte sunulmuştur.

• Bölüm 8, Modüler aritmetik kavramlarını içermektedir. Bu bölümde kalan sınıflar, asal kalan sınıflar ve sıfır bölen sınıflar konuları bazı GAP programı örnekleri ile birlikte sunulmuştur.

• Bölüm 9, Gruplar kavramını içermektedir. Bu bölümde bir yapının grup olup olmaması, grubun derecesi, elemanları, üreteçleri konuları ve bazı GAP programı örnekleri yer almaktadır.

• Bölüm 10, Alt grup kavramını içermektedir. Bu bölümde bir yapının alt grup olup olmaması, bir grubun merkezi, merkezleştirici, normalleştirici ve bir grubun mertebesi kavramları GAP programının `Subgroup()`, `IsSubgroup()`, `Center()`, `Centralizer()`, `Normalizer()` ve `Size()` fonksiyonarı ile test edilmiştir.

• Bölüm 11, Normal alt grup kavramını içermektedir. Bu bölümde bir grubun nomal alt gruplarının bulunması, denklik sınıfları, sağ yan cümle, has normal alt gruplar, Lagrange teoremi, indeks, Euler teoremi, Fermat teoremi, bölüm grubu, komütatör alt grup, basit grup, maksimal normal alt grup kavramları GAP programları örnekleri ile sunulmuştur.

• Bölüm 12, Grup homomorfizmleri kavramını içermektedir. Bu bölümde bir fonksiyonun homomorfizm olup olmaması, homomorfizmin tanım grubu, homomorfizmin değer grubu, görüntü, sıfır homomorfizm, birerib homomorfizm, örten homomorfizm ve çekirdek kavramları GAP programları örnekleri ile sunulmuştur.

• Bölüm 13, Simetrik gruplar kavramını içermektedir. Bu bölümde simetrik grup, permütasyon grubu, devir, özdeşlik fonksiyonu, ayrık devir, devirlerin bileşkesi, bir devrin mertebesi, eşlenik ve ayrışım kavramları GAP programları örnekleri ile sunulmuştur.

• Bölüm 14, Sylow teoremleri kavramını içermektedir. Bu bölümde p-sylow alt grubu, ve Sylow teoremleri GAP programının `SylowSubgroup()` fonksiyonu kullanılarak test edilmiş ve GAP programı örnekleri sunulmuştur.

• Bölüm 15, Halkalar kavramını içermektedir. Bu bölümde bir cebirsel yapının halka olup olmaması, dağılma özellikleri, halkanın abelyen olup olmaması, ters eleman, bir halkanın tersiner kümesi, alt halka kavramı, bir halkanın merkezi, idempotent ve nilpotent elemanlar ve

ideal kavramları GAP programı örnekleriyle sunulmuştur. Tamsayılar ve Rasyonel sayılar kümelerinin GAP programında kullanılmasıyla da ilk defa sonsuz kümeler için GAP programında işlemler yapılmıştır.

• Bölüm 16, Tamlık bölgesi ve cisim kavramlarını içermektedir. Bu bölümde sıfır bölen, tam halka, karakteristik, tamlık bölgesi, cisim, alt cisim, asal cisim ve Gaussian halkaları kavramları GAP programı örnekleriyle sunulmuştur.

• Bölüm 17, Halka homomorfizmleri kavramını içermektedir. Bu bölümde bir fonksiyonun halka homomorfizmi olup olmaması, sıfır homomorfizma, doğal homomorfizma, bir homomorfizmin görüntüsü ve çekirdeği kavramları GAP programı örnekleriyle sunulmuştur.

• Bölüm 18, 30. dereceye kadar olan bazı küçük grupların latislerini içermesinin yanı sıra grupların üreteçleri, elemanları, abelyenlik isimleri ve GAP programı sıralamaları bir liste halinde sunulmuştur.

2 GAP PROGRAMININ ELDE EDİLMESİ VE YÜKLENMESİ

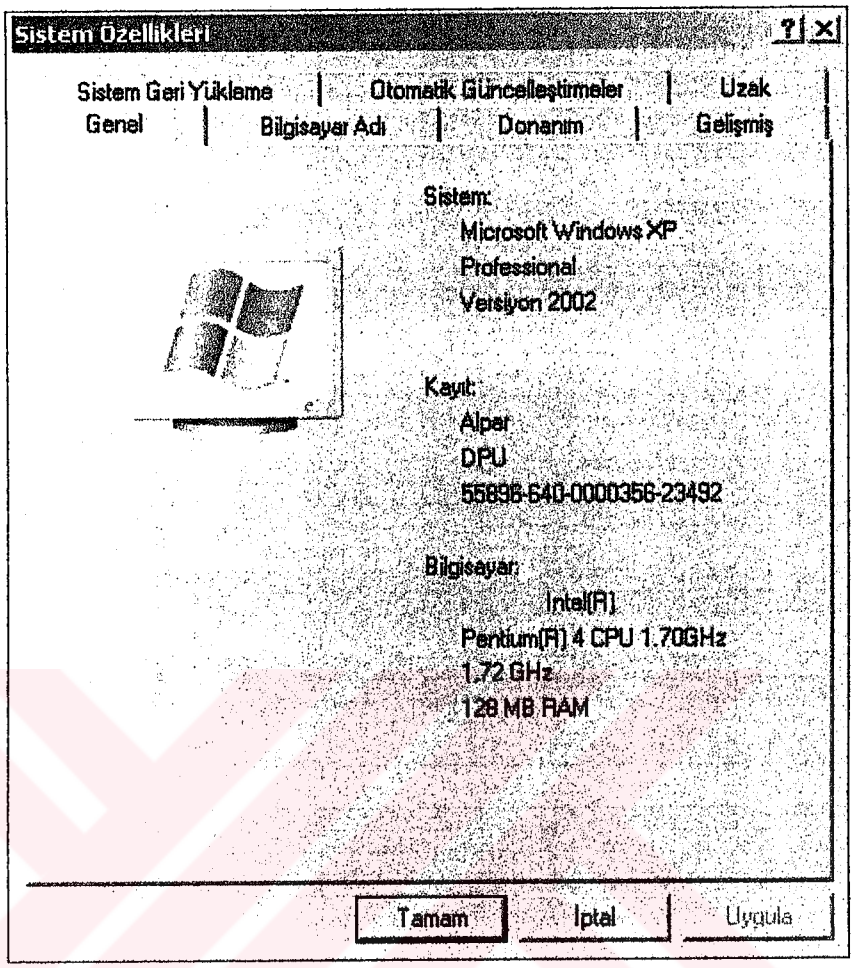
2.1 Giriş

Bu bölümde GAP programının bir bilgisayara yüklenmesi için gerekli olan bilgisayar donanımları ve GAP programının bilgisayara indirilmesi esnasında kullanılan formatlar adım adım açıklanmıştır. Programın bilgisayara kolay bir şekilde indirilmesi için var olan formatlar ve işletim sistemleri (Windows, Unix, Macintosh vb.) için programın elde edileceği adreslerde detaylı olarak verilmiştir.

Yine bu bölüm içerisinde elde edilen GAP programı Windows işletim sistemi kullanılarak adım adım yöntemi ile bir bilgisayara yüklenme aşamasını da içermektedir. Bir bilgisayara yüklenen GAP programının çalıştırılması için gerekli açıklamalar ve bilgisayarda masaüstüne kısayol oluşturma işlemleri de bu bölümde sunulmuş ve program çalışma aşamasına getirilmiştir. GAP programı çekirdeğinde var olan programlarla birlikte kütüphanesinde bulunan ortak paket programlarla da çalıştığından dolayı çekirdek dışında var olan paket programları da bu bölüm içerisinde paketin adı, adresi, konu başlığı ve paketi geliştiren bilim adamlarının açık bir şekilde verilmesiyle sunulmuştur. Bu bölümde kullanılan tüm GAP programı yükleme bilgileri [1] den alınmıştır.

2.2 Gerekli Bilgisayar Donanımı

Son versiyonu Mayıs 2002 olan GAP (4.3), internet üzerinde ücretsiz olarak kullanıma sahip bir programdır. GAP programının bir bilgisayarda çalışabilmesi için bilgisayarın en az 20 MB RAM e ve Pentium 133 MHz veya daha hızlı işlemciye sahip olması gerekir [1]. Bilgisayarınızın hızı ne kadar fazla olursa GAP penceresinin açılışı sırasında bekleme süreniz o kadar az olacaktır. Örneğin Intel Pentium 4 1.70 GHz işlemci ve 128 MB RAM a sahip bilgisayarda GAP programı tüm paketleri ile birlikte 15 saniye içerisinde yüklenip kullanıma hazır hale gelecektir. Bir bilgisayarın işlemcisinin hızı ve RAM miktarı, tüm Windows (95,98,98 Se, ME, 2000, XP) işletim sistemlerinde, Bilgisayarım simgesine farenin sağ tuşu tıklanarak öğrenilebilir.



Şekil 2.1 Bilgisayarın RAM ve işlemci bilgisini görüntüleme

GAP programı, tüm özellikleri ile birlikte yüklendikten sonra bir bilgisayarda yaklaşık 320 MB yer kaplayacaktır. GAP programının yükleme paketlerini bilgisayarınızda tutmak isterseniz bu dosyalar yaklaşık 80 MB yer kaplayacaktır [1].

GAP 4.3 UNIX, Windows veya Macintosh tabanlı bilgisayarlar üzerine kurulup çalıştırılabilmektedir. GAP programını yüklemek için, bilgisayarın niteliğine göre bir takım dosyalara ihtiyaç vardır.

2.3 GAP Programını Bilgisayara İndirmek

GAP programının bir bilgisayara yüklenebilmesi için arşiv dosyalarının o bilgisayara internet üzerinden indirilmesi gerekir. GAP programının yüklemesinin yapılabilmesi için mutlaka yükleme dosyalarının aşağıdaki üç formattan birisine ait olması gerekir [1].

.tar.gz formatı

UNIX/Linux tabanlı bilgisayarlar için kullanılan standart sıkıştırma formatıdır. (gunzip programı kullanılarak sıkıştırılmış dosyalar açılabilir.)

.zip formatı

Windows/DOS tabanlı bilgisayarlar için kullanılan standart sıkıştırma formatıdır. (Winzip veya winrar programları kullanılarak sıkıştırılmış dosyalar açılabilir.)

.zoo formatı

UNIX/Linux , Windows/DOS veya Macintosh tabanlı bilgisayarların hepsinde kullanılabilen bir formattır. (unzoo GAP programı için yazılmış bir yazılım olup unzoo kullanılarak sıkıştırılmış dosyalar açılabilir.)

Aşağıda verilen internet adreslerinden, GAP programını tüm kütüphaneleri ile birlikte yüklemek için, gerekli dosyalar bilgisayara indirilebilir.

2.3.1 UNIX/Linux

UNIX/Linux tabanlı bir bilgisayarda GAP programını yüklemek için aşağıdaki dosyalar, karşısında yer alan Internet adresinden indirilmelidir [1].

gap4r3.tar.gz ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/gap4r3.tar.gz
accpkg4r3.tar.gz ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.tar.gz
deppkg4r3.tar.gz ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/deppkg4r3.tar.gz

GAP programında meydana gelebilen bazı aksaklıkları düzeltmek için oluşturulmuş `fix` dosyaları GAP programının fonksiyonlarının kullanımı esnasında bir hata oluşturduğu takdirde `bugfixes` dizininde sürekli olarak güncellenmiş halde bulunabilir. Bu klasörün internet adresi de aşağıda verilmiştir.

bugfixes dizini ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bugfixes/

Ek işaret tablolarının GAP kütüphanesine eklenebilmesi için `xtom` dosyaları kullanılabilir. GAP programının çalışması için `xtom` dosyalarının kurulması şart değildir. Yani isteğe bağlıdır. Yüklenilmesi durumunda, ekstra olarak GAP programı, 80 MB daha fazla yer kaplayacaktır.

xtom4r3.tar.gz ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/xtom4r3.tar.gz

Yukarıda internet adresleri verilen arşiv dosyaları, GAP programının UNIX/Linux tabanlı bir bilgisayara kurulabilmesi için yeterli olmakla beraber, `zoo` uzantılı dosyalar da kullanılarak GAP programı yüklenebilir. Aşağıda internet adresleri verilen `zoo` dosyalarını açabilmek için `unzoo` programına ihtiyaç vardır. UNIX/Linux tabanlı bir bilgisayarda çalışabilecek `unzoo` sürümü aşağıdaki internet adresinden temin edilebilir.

unzoo.c ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/util/unzoo.c
accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
deppkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/deppkg4r3.zoo
xtom4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/xtom4r3.zoo

UNIX/Linux tabanlı bir bilgisayarda, arşiv dosyaların `tar.gz` yada `zoo` uzantılı olarak kullanılması GAP programı açısından bir fark oluşturmaz.

2.3.2 Windows

Windows tabanlı bir bilgisayarda GAP programını yüklemek için aşağıdaki dosyalar karşısında yer alan Internet adresinden indirilmelidir [1].

gappc4r3.zip ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/gappc4r3.zip
accpkg4r3.zip ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zip
deppkg4r3.zip ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/deppkg4r3.zip

GAP programında meydana gelebilen bazı aksaklıkları düzeltmek için oluşturulmuş `fix` dosyaları GAP programının fonksiyonlarının kullanımı esnasında bir hata oluşturduğu takdirde `bugfixes` dizininde sürekli olarak güncellenmiş halde bulunabilir. Bu klasörün internet adresi de aşağıda verilmiştir.

bugfixes dizini ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bugfixes/

Ek işaret tablolarının GAP kütüphanesine eklenebilmesi için `xtom` dosyaları kullanılabilir. GAP programının çalışması için `xtom` dosyalarının kurulması şart değildir. Yani isteğe bağlıdır. Yüklenilmesi durumunda, ekstra olarak GAP programı, 80 MB daha fazla yer kaplayacaktır.

xtom4r3.zip ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/xtom4r3.zip

Yukarıda internet adresleri verilen arşiv dosyaları GAP programının Windows tabanlı bir bilgisayara kurulabilmesi için yeterli olmakla beraber, `zoo` uzantılı dosyalar da kullanılarak GAP programı yüklenebilir. Aşağıda internet adresleri verilen `zoo` dosyalarını açabilmek için `unzoo` programına ihtiyaç vardır. Windows tabanlı bir bilgisayarda çalışabilecek `unzoo` sürümü aşağıdaki internet adresinden temin edilebilir.

unzoo.exe ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bin/win/unzoo.exe
accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
deppkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/deppkg4r3.zoo
xtom4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/xtom4r3.zoo

Windows tabanlı bir bilgisayarda arşiv dosyaların `tar.gz` yada `zoo` uzantılı olarak kullanılması GAP programı açısından bir fark oluşturmaz.

2.3.3 Macintosh

Macintosh tabanlı bir bilgisayarda GAP programını yüklemek için aşağıdaki dosyalar karşısında yer alan Internet adresinden indirilmelidir [1].

accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
accpkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/accpkg4r3.zoo
deppkg4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/deppkg4r3.zoo

GAP programında meydana gelebilen bazı aksaklıkları düzeltmek için oluşturulmuş `fix` dosyaları GAP programının fonksiyonlarının kullanımı esnasında bir hata oluşturduğu takdirde `bugfixes` dizininde sürekli olarak güncellenmiş halde bulunabilir. Bu klasörün internet adresi de aşağıda verilmiştir.

bugfixes dizini ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bugfixes/

Ek işaret tablolarının GAP kütüphanesine eklenebilmesi için `xtom` dosyaları kullanılabilir. GAP programının çalışması için `xtom` dosyalarının kurulması şart değildir. Yani isteğe bağlıdır. Yüklenilmesi durumunda, ekstra olarak GAP programı, 80 MB daha fazla yer kaplayacaktır.

xtom4r3.zoo ---> ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/xtom4r3.zoo

Yukarıda internet adresleri verilen zoo dosyalarını açabilmek için unzoo programına ihtiyaç vardır. Macintosh tabanlı bir bilgisayarda çalışabilecek unzoo sürümü aşağıdaki internet adresinden temin edilebilir.

unzoo4r3n1-PPC.sit veya unzoo4r3-68k.sit programları Macintosh için kullanılan unzoo sürümleridir.

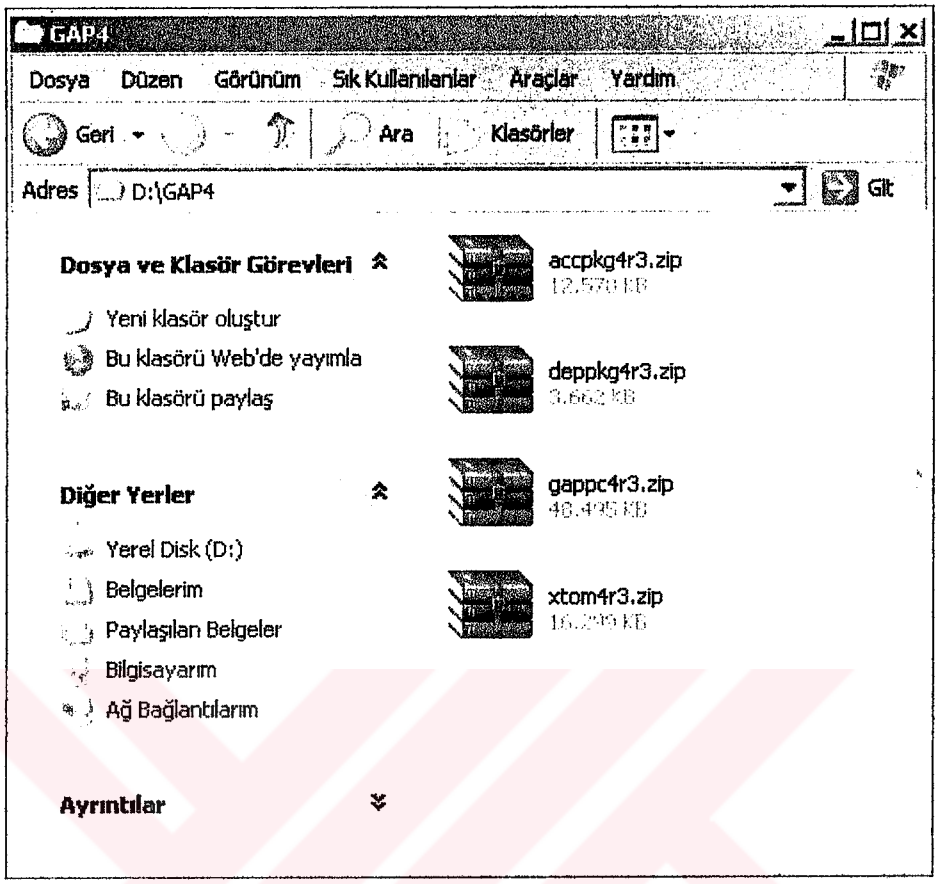
ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bin/mac/unzoo4r3n1-PPC.sit
ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bin/mac/unzoo4r3-68k.sit

2.4 GAP Programını Bilgisayara Kurmak

2.4.1 Windows

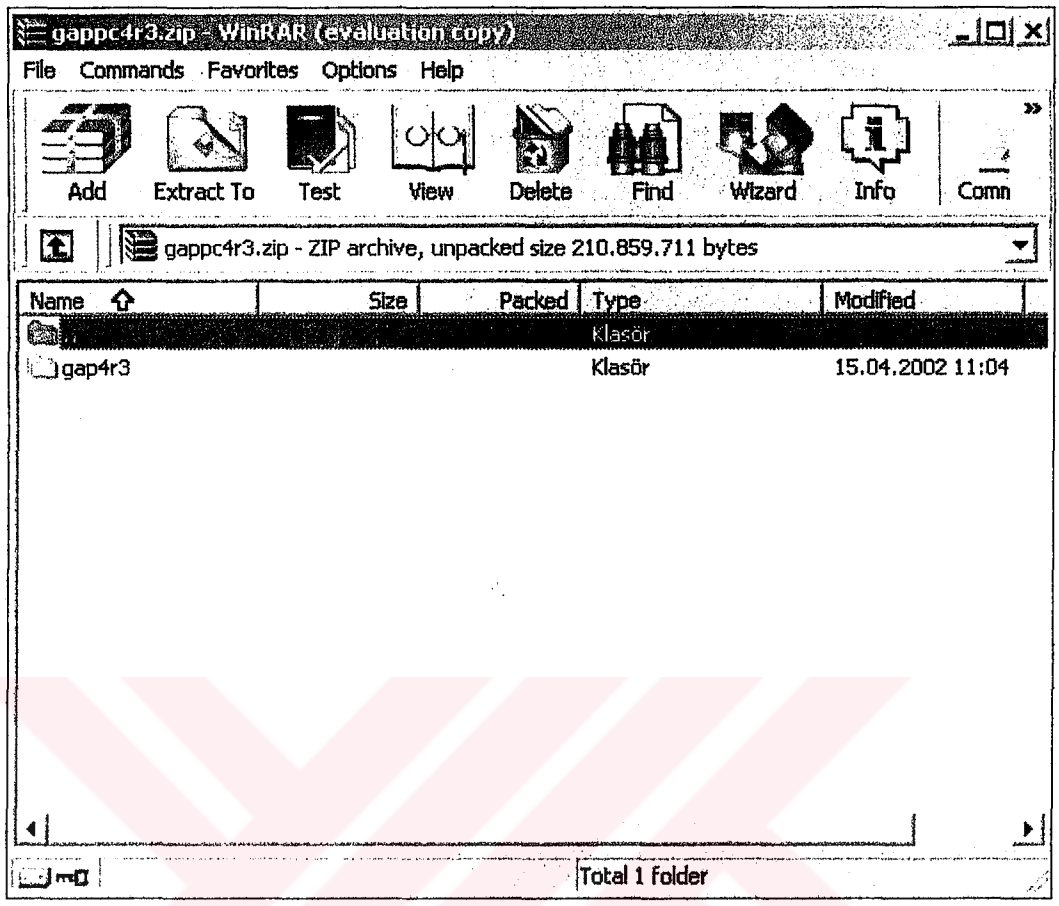
Windows tabanlı bir bilgisayarda, zip uzantılı arşiv dosyaları kullanılarak GAP programı yüklenebilir. GAP programının yüklemesini gerçekleştirmek için deneme, sürümü Internet üzerinden ücretsiz olarak yüklenebilen, WinRar programı kullanılacaktır.

Bilgisayara indirilen gappc4r3.zip, accpkg4r3.zip ve deppkg4r3.zip dosyaları, Windows tabanlı bir bilgisayara GAP programını yüklemek için gerekli zip uzantılı arşiv dosyalarıdır. Bu dosyalarla birlikte xtom4r3.zip arşiv dosyasını da bilgisayara kaydedilmelidir. Bilgisayarda WinRar programı yüklü ise zip uzantılı arşiv dosyaları WinRar programının şeklini alacaktır.



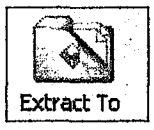
Şekil 2.2 GAP programı yükleme dosyaları

GAP programını bilgisayarın C:\ sürücüsüne C:\gap4r3 isimli dizine kurmalıyız. İlk önce gappc4r3.zip dosyası WinRar kullanarak açılır. WinRar programının görüntüsü aşağıdaki gibi olacaktır.



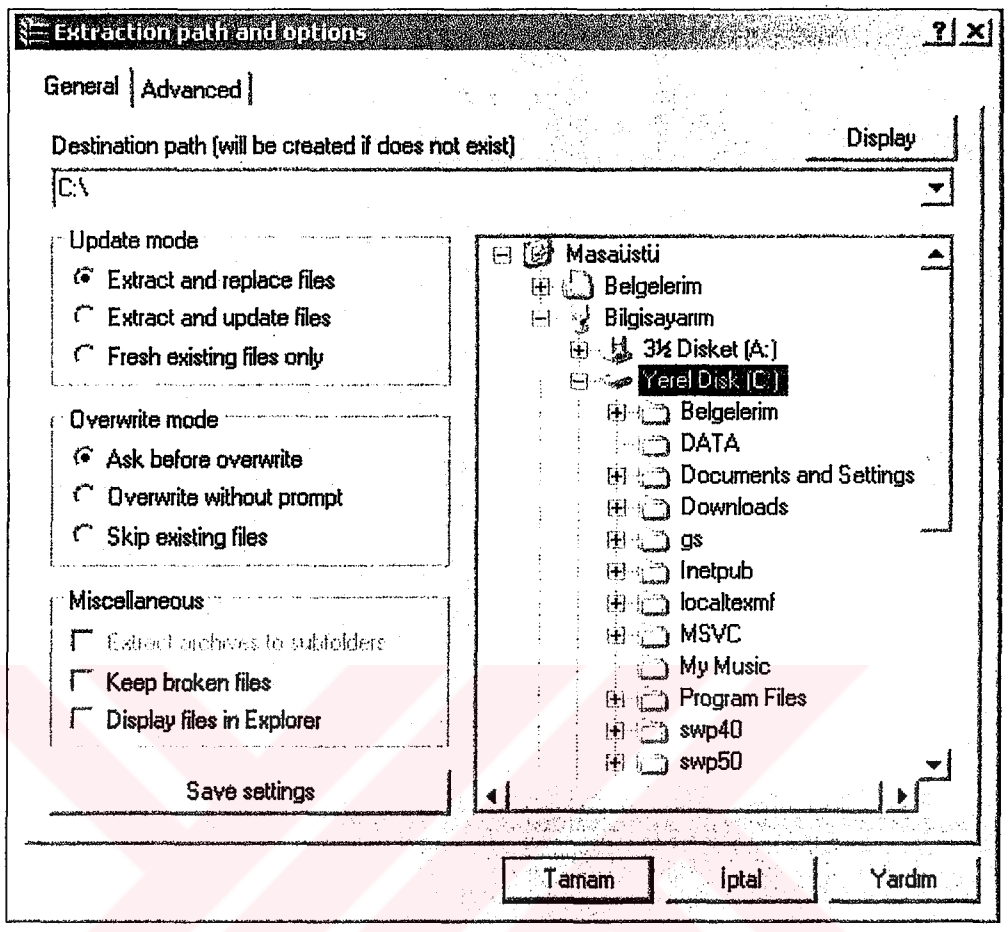
Şekil 2.3 WinRar programı görüntüsü

Bu pencerede yer alan **Extract To** simgesine basılırsa bu sıkıştırılmış dosyanın açılmak istenilen yer belirtilir.



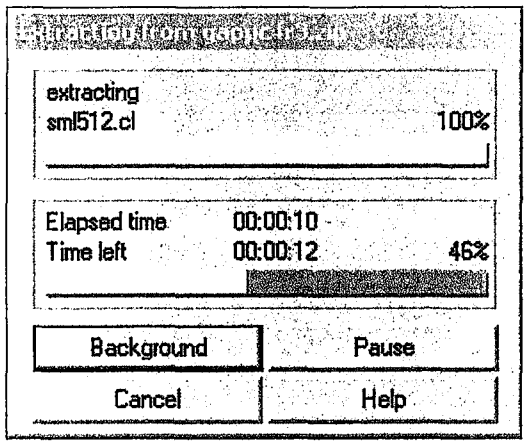
Şekil 2.4 Winrar programı "Extract To" simgesi

C:\ sürücüsü içersine açılacak bu arşiv dosyası C:\gap4r3 dizinini oluşturacak ve içerisindeki dosyaları bu dizine kopyalayacaktır. Bu işlemden sonra gappc4r3.zip arşiv dosyası açılmış olur.



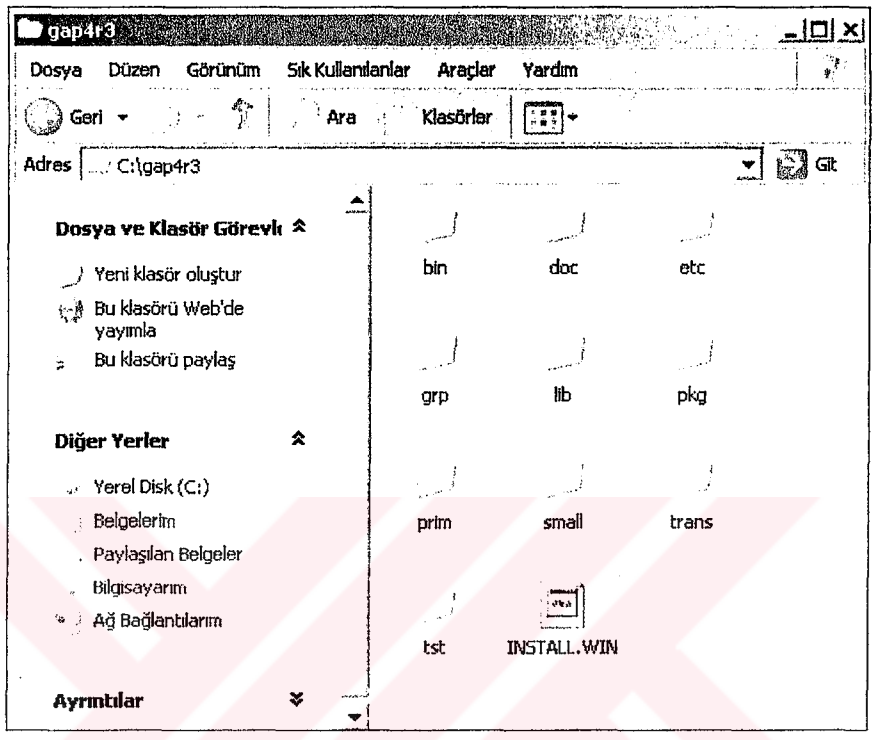
Şekil 2.5 GAP programının yükleneceği dizinin ayarlanması

Yukarıdaki pencerede kopyalanacak yer olarak C:\ seçilip **Tamam** tuşuna basıldığında kopyalama işlemi başlayacaktır.



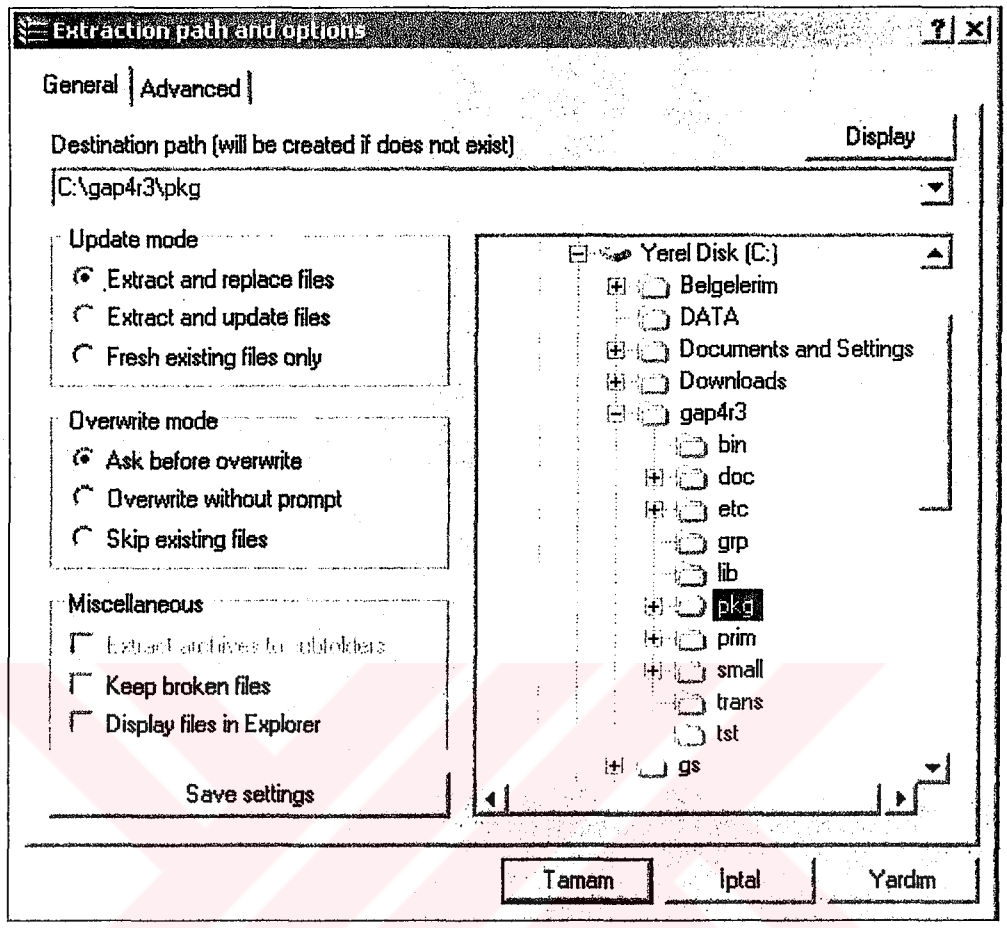
Şekil 2.6 Arşiv dosyasının genişletilmesi

C:\gap4r3 dizini, GAP programı için kök dizini yani ana dizin olacak ve içerisinde aşağıdaki gibi klasörler oluşmuş olacaktır.



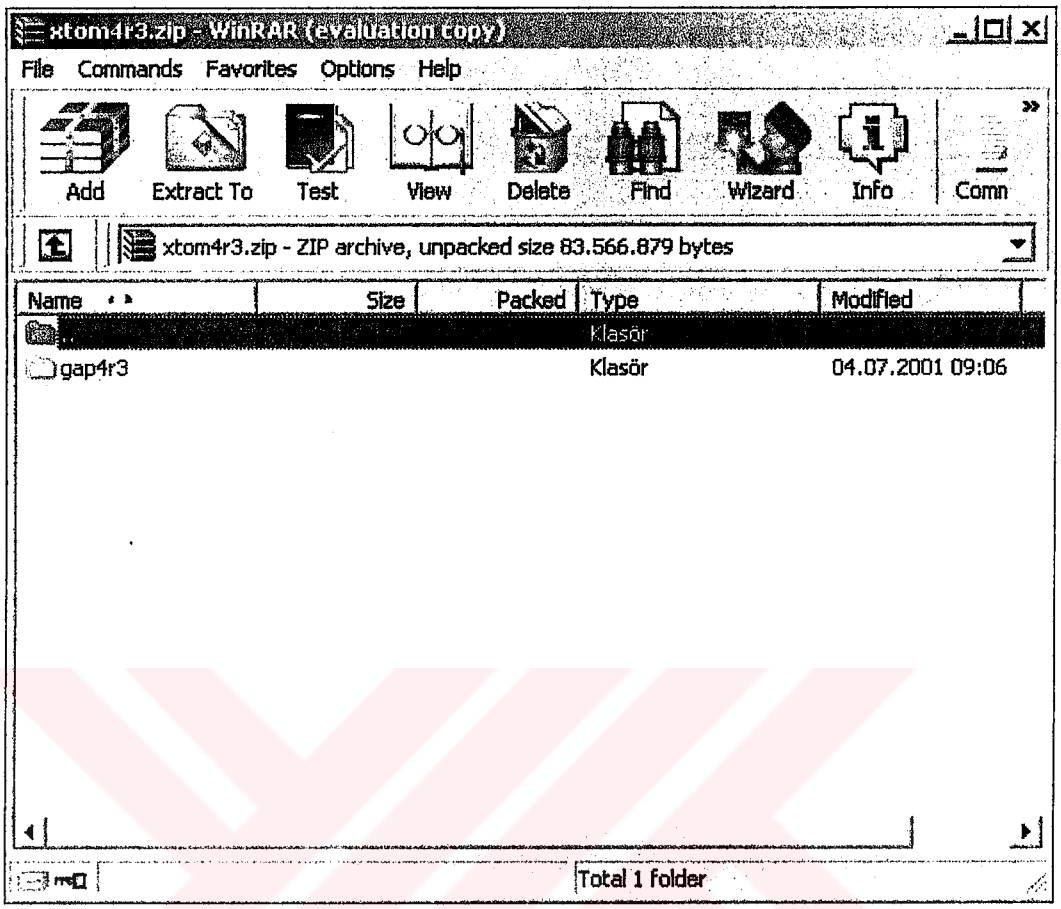
Şekil 2.7 GAP programı ana dizini

Daha sonra accpkg4r3.zip ve deppkg4r3.zip arşiv dosyalarını, bu sırayla C:\gap4r3 dizini içerisindeki pkg dizinine WinRar kullanılarak açılır. WinRar programı kullanılarak accpkg4r3.zip arşiv dosyası açılmalı ve **Extract To** tuşuna basılarak bu arşiv dosyası açılmak istenilen yer olarak C:\gap4r3\pkg dizinini göstererek **Tamam** tuşuna basılmalıdır. Bu işlemden sonra accpkg4r3.zip arşiv dosyasının içersindeki dosyalarda genişletilmiş olacaktır. Aynı işlem deppkg4r3.zip arşiv dosyası içinde uygulanarak bu dosya içerisindeki dosyaları da C:\gap4r3\pkg klasörü içerisine açılır.



Şekil 2.8 Arşiv dosyaları için dizin hazırlama

Buradaki gappc4r3.zip, accpkg4r3.zip ve deppkg4r3.zip arşiv dosyalarının bu şekildeki açılma işlemleri tamamladıktan sonra GAP programı yüklenmiş olur. Buradaki xtom4r3.zip arşiv dosyası, kullanıcının isteğine bağlı olarak kullanılabilir. Bu dosya da kullanılmak istenirse yine WinRar programı ile birlikte **Extract To** tuşunu kullanarak C:\ dizini içerisine açılmalıdır. Bu işlem, GAP programı için bilgisayarda fazladan 80 MB daha yer ayrılmasını gerektirecektir.

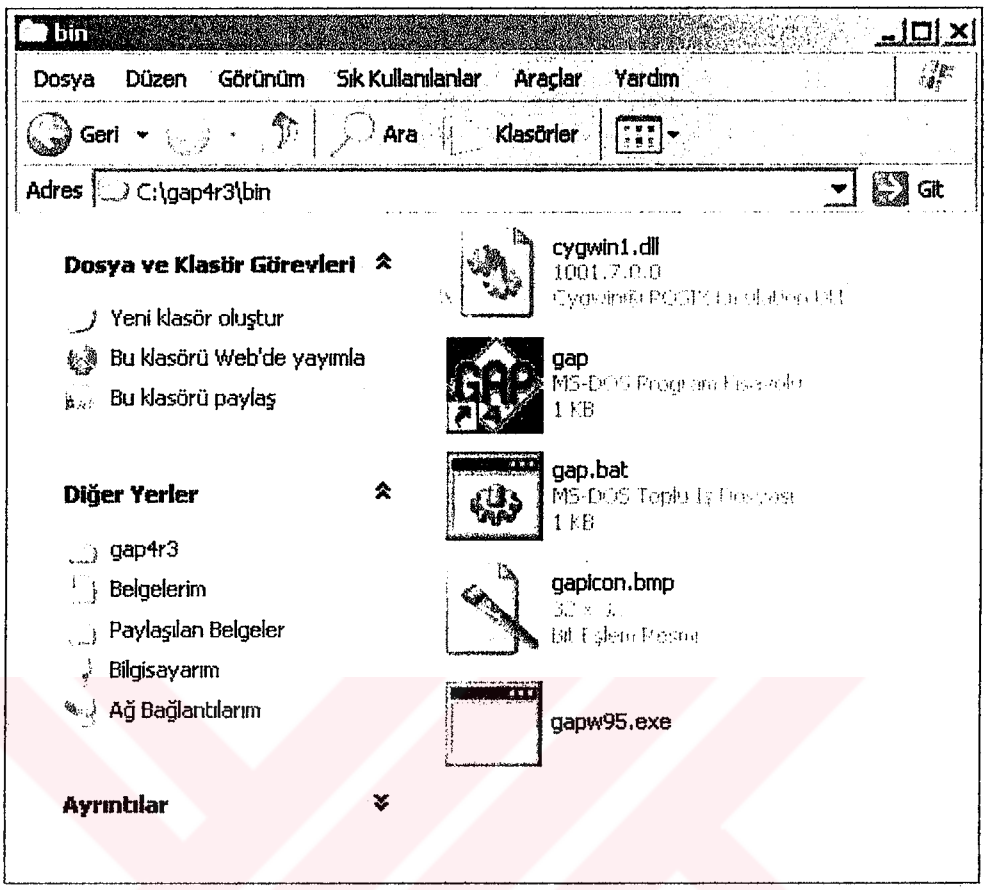


Şekil 2.9 Ek arşiv dosyalarının genişletimesi

ftp://ftp-gap.dcs.st-and.ac.uk/pub/gap/gap4/bugfixes/ ftp adresine bağlanarak bugfixes dizininden yeni güncelleştirmelerin indirilmesi halinde bu arşiv dosyaları da, yine WinRar programı kullanılarak C:\ dizini içerisine açılmalıdır.

2.5 GAP Programını Çalıştırmak ve Masaüstüne Kısayol Oluşturmak

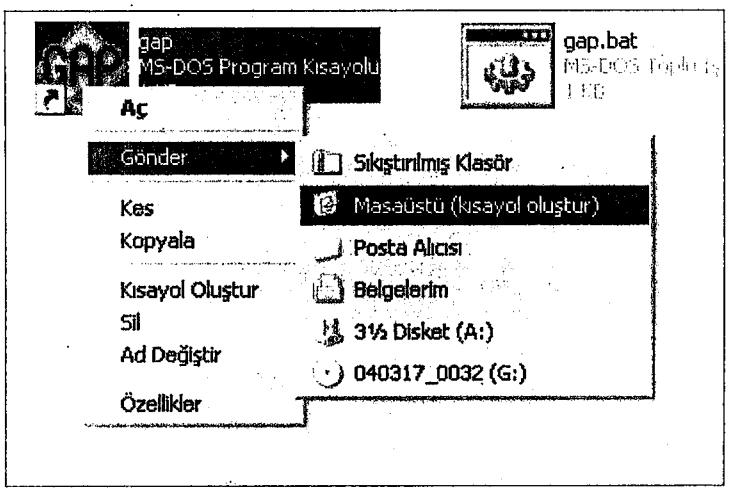
C:\gap4r3 dizini, GAP programının yüklü olduğu ana klasördür. GAP programını çalıştırmak için C:\gap4r3\bin klasörüne gidildiğinde, bu klasör içersinde beş adet dosyanın var olduğu görülecektir.



Şekil 2.10 GAP programı C:\bin dizini

Şekil 2.11 GAP programı simgesi

C:\gap4r3\bin dizini içerisinde yer alan yukarıdaki simge tıklatılarak GAP programı çalıştırılabilir. Bu simge aslında `gap.bat` dosyasının çalıştırılabilmesi için bir kısa yoldur. GAP programını her defasında C:\gap4r3\bin dizini içerisine girip çalıştırmak yerine buradaki kısa yol simgesinden masaüstüne kopyalayarak kısa yol oluşturulabilir. Bunu yapmak için bu belge üzerine gelinip farenin sağ tuşu ile tıklanır. Açılan menüde **Gönder** simgesinin üzerine geldiğimizde aşağıdaki şekildeki gibi yeni bir menü açılacaktır. Bu menüden **Masaüstü (kısayol oluştur)** simgesine tıklanarak masaüstüne bir kısayol yerleştirilir.



Şekil 2.12 Masaüstüne kısayol oluşturulması

Masaüstüne yerleştirilen simgeye farenin sol tuşu ile çift tıklanarak GAP programı çalıştırılır.

Bilgisayarın RAM ve işlemci durumuna göre kısa bir süre sonra GAP programı kullanıma hazır hale gelir. GAP programının kullanıma hazır hale geldiği en alt satırda yer alan `gap>` promptunun gelmesi ve imlecin yanıp sönmesinden anlaşılır.

Şekil 2.13 GAP programı görüntüsü

2.6 GAP Programı Ortak Paketleri

- Paketin Adı : GRAPE
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/grape.html
  Konusu : GRAPE for constructing and analysing graphs related to groups, finite geometries, and designs.
  Yapan : Leonard Soicher

- Paketin Adı : cohomolo
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/cohomolo.html
  Konusu : cohomolo for computing Schur multipliers and covering groups of finite groups and first and second cohomology groups of finite groups acting on finite modules.
  Yapan : Hans Derek F. Holt

- Paketin Adı : LAG
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/lag.html
  Konusu : Grup cebirlerinin Lie cebirleri.
  Yapan : Richard Rossmanith
  Editörü : Andrea Caranti (Povo (Trento)), March 1999

- Paketin Adı : LAGUNA
  İnternet Adresi : http://ukrgap.exponenta.ru/laguna.htm
  Konusu : Lie AlGebras and UNits of group Algebras.
  Yapan : Victor Bovdi, Alexander Konovalov, Richard Rossmanith and Csaba Schneider.
  Editörü : H. Pahlings (Aachen), June 2003

- Paketin Adı : FactInt
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/factint.html
  Konusu : Integer Factorization
  Yapan : Stefan Kohl
  Editörü : Mike Atkinson (Dunedin), July 1999

- Paketin Adı : FarGAP
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/pargapmpi.html
  Konusu : Parallel GAP, Communication between a master GAP and several slave GAP's on other machines, (UNIX only)
  Yapan : Gene Cooperman
  Editörü : Steve Linton (St. Andrews), July 1999.

- Paketin Adı : XGAP
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/xgap.html
  Konusu : XGAP for GAP4 - a graphical user interface, (UNIX only)
  Integer Factorization
  Yapan : Frank Celler and Max Neunhöffer
  Editörü : Gerhard Hill (Aachen), July 1999.

- Paketin Adı : GrpConst
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/grpconst.html
  Konusu : Construction of all groups of a given order.
  Yapan : Hans Ulrich Besche and Bettina Eick.
  Editörü : Charles Wright (Eugene), July 1999

- Paketin Adı : FPLSA
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/fplsa.html
  Konusu : Interface to fast external Lie Todd-Coxeter Program.
  Yapan : Vladimir Gerdt and Vladimir Kornyak.
  Editörü : Steve Linton (St. Andrews), July 1999

- Paketin Adı : EDIM
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/edim.html
  Konusu : Interface to fast external Lie Todd-Coxeter Program.
  Yapan : Frank Lübeck
  Editörü : Atkinson (Dunedin), August 1999

- Paketin Adı : Cryst
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/cryst4.html
  Konusu : Bettina Eick, Franz Gähler and Werner Nickel.
  Yapan : Frank Lübeck
  Editörü : Herbert Pahlings (Aachen), February 2000.

- Paketin Adı : CrystCat
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/crystcat.html
  Konusu : A catalog of crystallographic groups of dimensions 2, 3, and 4.
  Yapan : Volkmar Felsch and Franz Gähler.
  Editörü : Herbert Pahlings (Aachen), February 2000.

- Paketin Adı : Carat
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/carat.html
  Konusu : Interface to Carat.
  Yapan : Volkmar Felsch and Franz Gähler.
  Editörü : Herbert Pahlings (Aachen), February 2000.

- Paketin Adı : ITC
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/itc.html
  Konusu : Interactive Todd-Coxeter (UNIX only).
  Yapan : Volkmar Felsch, Ludger Hippe and Joachim Neubüser.
  Editörü : E. Robertson (Dunedin), March 2000.

- Paketin Adı : AutPGrp
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/autpgrp.html
  Konusu : Automorphism Group of a p-Group.
  Yapan : Bettina Eick and Eamonn O'Brien.
  Editörü : Derek F. Holt (Warwick), September 2000

- Paketin Adı : CRISP
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/crisp.html
  Konusu : Computing with Radicals, Injectors Schunck classes and Projectors of finite solvable groups.
  Yapan : Burkhard Höfling.
  Editörü : Joachim Neubüser (Aachen), December 2000.

- Paketin Adı : CRISP
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/crisp.html
  Konusu : Computing with Radicals, Injectors Schunck classes and Projectors of finite solvable groups.
  Yapan : Burkhard Höfling.
  Editörü : Joachim Neubüser (Aachen), December 2000.

- Paketin Adı : FORMAT
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/format.html
  Konusu : Computing with formations of finite solvable groups.
  Yapan : Bettina Eick and Charles Wright.
  Editörü : Joachim Neubüser (Aachen), December 2000.

- Paketin Adı : ACLIB
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/aclib.html
  Konusu : Almost Crystallographic Groups Library.
  Yapan : Bettina Eick and Charles Wright.
  Editörü : Gerhard Hill (Aachen), February 2001.

- Paketin Adı : AtlasRep
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/atlasrep.html
  Konusu : Atlas of Group Representations.
  Yapan : Robert Wilson, Richard A. Parker, John Bray and Thomas Breuer.
  Editörü : Herbert Pahlings (Aachen), April 2001.

- Paketin Adı : ACE
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/ace.html
  Konusu : Advanced Coset Enumerator.
  Yapan : George Havas, Colin Ramsay, Alexander Hulpke and Greg Gamble.
  Editörü : Joachim Neubüser (Aachen), April 2001.

- Paketin Adı : SmallGroups
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/sgl.html
  Konusu : Small Group Library.
  Yapan : Hans Ulrich Besche, Bettina Eick and Eamonn O'Brien
  Editörü : Mike F. Newman (Canberra), June 2001.

- Paketin Adı : ANUPQ
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/anupq.html
  Konusu : ANU p-quotient.
  Yapan : Eamonn O'Brien, Werner Nickel and Greg Gamble.
  Editörü : Charles Wright (Eugene), April 2002.

- Paketin Adı : XMod
  İnternet Adresi : http://www-gap.dcs.st-and.ac.uk/~gap/Share/xmod4.html
  Konusu : Crossed Modules and cat^1 groups.
  Yapan : Chris Wensley and Murat Alp.
  Editörü : Derek F. Holt (Warwick), December 1996.

- Paketin Adı : NQ
  İnternet Adresi : http://www.mathematik.tu-darmstadt.de/~nickel/software/NQ/
  Konusu : ANU Nilpotent Quotient Algorithm.
  Yapan : Werner Nickel.
  Editörü : Joachim Neubüser(Aachen), January 2003.

- Paketin Adı : GUAVA
  İnternet Adresi : http://web.usna.navy.mil/~wdj/gap/GAP_package_GUAVA.html
  Konusu : Coding Theory.
  Yapan : R. Baart, J. Cramwinckel, E. Roijackers, E. Minkes, Lea Ruscio.
  Editörü : David Joyner.

- Paketin Adı : KBMAG
  İnternet Adresi : http://www.maths.warwick.ac.uk/~dfh/kbmag/
  Konusu : Knuth-Bendix for Monoids and Groups.
  Yapan : Derek F. Holt.
  Editörü : Charles Wright (Eugene), September 2003.

- Paketin Adı : QuaGroup
  İnternet Adresi : http://www.math.uu.nl/people/graaf/quagroup.html
  Konusu : Computing with quantized enveloping algebras.
  Yapan : Willem de Graaf.
  Editörü : Gerhard Hill (Aachen), September 2003.

3 GAP PROGRAMLAMA DİLİ

3.1 Giriş

GAP programı Pascal ve C programlama dilleri ile yazılmış bir paket programdır. Pascal ve C kodları ortak kullanılarak, yalnızca GAP programı içerisinde kullanılabilen ve yapı olarak daha çok Pascal a benzeyen yeni bir programla dili oluşturulmuştur. Diğer programlama dilleri gibi değişkenlere, sabitlere, denetim deyimlerine, aritmetik operatörlere ve döngü deyimlerine sahip olan bu programlama dili GAP programı içersinde kullanılabilir ve yorumlanabilir hale getirilmiştir.

Bu nedenle bu bölüm içerisinde GAP programında kullanılacak semboller, anahtar kelimeler, değişken isimleri, deyimler ve açıklamaların nasıl yapılacağı, değişken ve değişken atama yöntemleri, karşılaştırma operatörleri (<,>,<=,>=,<>,=), aritmetik operatörler, mantıksal operatörler çeşitli örneklerle detaylı bir şekilde sunulmuştur. Ayrıca bu bölüm içerisinde GAP programı denetim deyimleri `if`, `while`, `repeat`, `for`, `break`, `continue`, `print` deyimleri de çeşitli basit örneklerle sunulmuştur.

3.2 Semboller

GAP programlama dili içerisinde kullanılan özel tanımlı karakter ve semboller vardır. Aşağıdaki semboller, aritmetik işlemler, karşılaştırmalar gibi birçok işlevi gerçekleştirmek için kullanılırlar [17].

```
+    -    *    /    ^    ~    !.
=    <>   <    <=   >    >=   ![
:=   .    ..   ->   ,    ;    !{
[    ]    {    }    (    )    :
```

3.3 Anahtar Kelimeler

GAP programlama dili içerisinde, kullanılacak fonksiyonlar için ayrılmış özel kelimeler vardır. Aşağıda listesi verilen anahtar kelimeler, değişken ismi olarak kullanılamazlar [17].

```
and       do        elif    else     end      fi
for       function  if      in       local    mod
not       od        or      repeat   return   then
until     while     quit    QUIT     break    rec
continue
```

GAP programı kullanıcısı bu anahtar kelimeleri değişken ismi olarak kullanırsa bir hata mesajı ile karşılaşacaktır.

```
gap> and:=2;
Syntax error: expression expected
and:=2;
  ^
```

GAP programı büyük harf küçük harf duyarlılığına sahiptir. Hemen hemen tüm anahtar kelimelerin küçük harfler kullanılarak yazıldığına dikkat edilmelidir. Örneğin `while` anahtar kelimesindeki harflerden herhangi birisi büyük harfle yazılırsa bu kelime anahtar kelime olma özelliğini kaybeder. `While` , `wHile` , `whIle` , `whiLe` ve `whilE` kelimeleri anahtar kelime değildirler ve değişken ismi olarak kullanılabilirler. Anahtar kelimeler arasında boşluk kullanılmamalıdır. `el if` şeklindeki bir yazılım `elif` anahtar kelimesinin yaptığı işlevi yerine getirmeyecektir [17].

Yukarıdaki anahtar kelimelere ek olarak aşağıdaki kelimeler de GAP programlama dili tarafından anahtar kelime olarak ayrılmıştır ve değişken ismi olarak kullanılamazlar.

```
false       true      IsBound        Unbind     TryNextMethod
Info        Assert    SaveWorkspace  fail
```

3.4 Değişken İsimleri

Bir değişkene isim atamak için değişken isimleri kullanılır. Değişken isimleri tanımlanırken anahtar kelimelerden farklı olmak üzere büyük harfler, küçük harfler, rakamlar ve alt çizgi (_) işareti kullanılabilir. GAP programında değişken isimleri tanımlanırken harfler

kullanılacaksa mutlaka İngilizce karakterler kullanılmalıdır. GAP programında büyük, küçük harf duyarlılığı olduğundan aynı kelimenin büyük harflerle veya küçük harflerle yazılması farklı değişken isimlerine karşılık gelecektir. Aşağıda verilen değişken isimleri farklı değişkenlere karşılık gelirler:

```
Degisken1       degisken1       dEgisken1       DEGISKEN1
Degisken_1      degisken_1      dEgisken_1      DEGISKEN_1
1_Degisken      1_degisken      1_dEgisken      1_DEGISKEN
_1Degisken      _1degisken      _1dEgisken      _1DEGISKEN
```

Değişken isimleri tanımlanırken herhangi bir karakter uzunluğu sınırlaması olmamasına rağmen GAP programı için yalnızca ilk 1023 karakter anlamlıdır. GAP programı için ilk 1023 karaktere kadar aynı olan iki değişken isimleri 1023 ten sonraki karakterler farklı olsa dahi aynı değişkeni gösterecektir.

3.5 Deyimler ve Açıklama

Bir GAP programı komutunda sağ taraftaki kısım deyim olarak adlandırılır. Deyimler değer yapılarını ifade etmekte kullanılırlar. Genel olarak deyimler bir değer olmayıp bir veya birkaç tane değerin aritmetik operatörlerle veya fonksiyonlarla bir işlem gerçekleştiren yapılardır. Örneğin 1+12 bir deyim, bu deyimin sonucu olan 13 bir değerdir. GAP programında # işareti bir açıklama yapmak için kullanılır [17].

```
gap> 2+3; # basit bir toplama islemi
5
gap> 2*3; # basit bir carpma islemi
6
```

3.6 Değişkenler

Diğer programlama dillerinde olduğu gibi GAP programında da değişkenler, bir değeri tutmak için kullanılırlar. Her bir değişken ismine bir tanımlayıcı (identifier) denir. GAP programında değişken tanımlama işlemi PASCAL programlama diline benzer. Bir tanımlayıcı (identifier) genel olarak `tanımlayıcı_ismi:=değer;` biçiminde oluşturulur. Bir ifade tanımlayıcı ismi olarak tanımlanmamışsa bu değerin deyim olarak kullanılması GAP programının hata vermesine sebep olacaktır. Daha önce oluşturulmuş bir tanımlayıcı (identifier) için yeni bir değer girildiğinde, tanımlayıcı artık o değeri kullanmaya başlar [17].

```
gap> a:=16;
16
gap> a;
16
gap> b;
Variable: 'b' must have a value
gap> a:=17;
17
gap> a;
17
```

GAP programında değişkenlerin büyük çoğunluğu işlevlere erişmek için kullanılırlar. GAP programında tanımlayıcı (identifier) ismi olarak anahtar kelimeler kullanılmaz. Anahtar kelimelerin dışında GAP programı içerisinde kullanmak istenilen tanımlayıcı (identifier) isimleri, GAP programı kütüphanesinde bir fonksiyon olarak tanımlanmışsa bu isimler de kullanılamaz GAP programı bu değerin salt okunur olduğunu belirtip hata verecektir.

GAP programında `IsReadOnlyGlobal()` komutu, değişken ismi olarak kullanmak istenilen karakterlerin salt okunur olup olmadığını yani GAP programı içerisinde bir fonksiyon ismi olarak kullanılıp kullanılmadığını test eder. `()` işaretleri arasına salt okunur olup olmadığı test edilmek istenen karakterler `"` işaretleri arasında yazılır. GAP programı `true` veya `false` yanıtı verecektir. Bu fonksiyonun sonucu olarak `true` yanıtı gelirse, salt okunur olup olmadığı test edilen karakterlerin tanımlayıcı (identifier) ismi olarak kullanılamayacağı açıktır [17].

```
gap> IsReadOnlyGlobal("Group");
true
gap> Group:=2;
Variable: 'Group' is read only
not in any function
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' after making it writable to continue
brk>
```

Yukarıdaki örnekte salt okunur olup olmadığı sorgulanan `Group` karakteri, grup cebirsel yapılarını oluşturmak için yazılmış bir fonksiyon olup, tanımlayıcı ismi olarak kullanılamaz.

GAP programı kullanılırken `MakeReadOnlyGlobal()` komutunu kullanılarak tanımlayıcı ismi olarak kullanılmaması istenilen herhangi bir karakter salt okunur hale getirilebilir. Bir karakter değişken ismi olarak atandıktan sonra bu komut kullanılırsa, artık bu değişken ismine yeni bir değer atayamayız. ve sürekli sabit kalır. Burada dikkat edilecek olan nokta GAP programı kapatılıp yeniden açıldığında bu komutla salt okunur hale getirilen karakterler bu özelliklerini kaybedeceklerdir. `()` işaretleri arasına, salt okunur hale getirilmek istenen karakterler `"` işaretleri arasında yazılır. Salt okunur hale getirmek istenilen karakterler zaten salt okunur özellikte ise GAP programı hata verecektir.

```
gap> pi:=22/7;
22/7
gap> MakeReadOnlyGlobal("pi");
gap> pi:=4;
Variable: 'pi' is read only
not in any function
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' after making it writable to continue
brk> quit;
gap> pi;
22/7
```

GAP programı içerisinde fonksiyon olarak kullanılan karakterleri yada `MakeReadOnlyGlobal()` komutunu kullanılarak salt okunur hale getirilen karakterler, tanımlayıcı ismi olarak kullanılmak istenirse `MakeReadWriteGlobal()` komutunu kullanarak `()` işaretleri arasına `"` işaretleri arasında yazılan karakterler yazılıp okunabilir hale getirilebilir.. Yazılabilir hale getirilmek istenen karakterler zaten yazılabilir özellikte ise GAP programı hata verecektir [17].

```
gap> IsReadOnlyGlobal("pi");
true
gap> MakeReadWriteGlobal("pi");
```

```
gap> pi:=4;
4
gap> IsReadOnlyGlobal("pi");
false
gap> MakeReadWriteGlobal("Group");
gap> Group:=2;
2
```

GAP programında bir tanımlayıcı isminin hangi değeri tuttuğunu öğrenmek için `tanımlayıcı_ismi;` komutunu kullanmak yeterlidir. `ValueGlobal()` komutu da yine bir tanımlayıcı isminin tuttuğu değeri öğrenmek için kullanılabilir [17].

```
gap> pi;
4
gap> ValueGlobal("pi");
4
```

Bir değişken isminin GAP programında bir değer tutup tutmadığı `IsBoundGlobal()` komutu kullanılarak test edilebilir [17].

```
gap> IsBoundGlobal("pi");
true
gap> IsBoundGlobal("k");
false
```

GAP `UnbindGlobal()` şeklinde verilen bir komut daha önce tanımlayıcı olarak atanan bir değeri, tanımlayıcı olmaktan çıkarıp tekrar boş karakter haline dönüştürür. () işaretleri arasına tanımlayıcı olmaktan çıkarıp tekrar boş hale getirilmek istenen karakterler "" işaretleri arasında yazılır [17].

```
gap> UnbindGlobal("pi");
gap> pi;
Variable: 'a' must have a value
```

GAP programında bir tanımlayıcı isminin genel olarak `tanımlayıcı_ismi:=değer;` biçiminde oluşturulduğundan bahsedilmişti.

BindGlobal() komutu da bir tanımlayıcı ismi oluşturmak için kullanılabilir. () işaretleri arasına , "" işaretleri arasında tanımlayıcı ismini oluşturacak karakter ve , işareti kullanılarak bu karakterin tutacağı değer yazılmalıdır. Bu komut kullanılarak oluşturulan tanımlayıcı isimleri otomatik olarak salt okunur olarak ayarlanacaktır. Tüm çalışma boyunca sabit kalması istenilen değerler için bu komut kullanılabilir.

```
gap> D:=20;
20
gap> BindGlobal("D",25);
#W BIND_GLOBAL: variable `D' already has a value
gap> D;
25
gap> IsReadOnlyGlobal("D");
true
```

NamesSystemGVars() komutu, GAP programı çalıştırılmaya başlandığında GAP programı kütüphanesi tarafından oluşturulan tüm değişken isimlerini listelemek için kullanılır.

NamesUserGVars() komutu, GAP programı çalıştırılmaya başlandığında, kullanıcı tarafından oluşturulan tüm değişken isimlerini listelemek için kullanılır [17].

```
gap> NamesUserGVars();
[ "A", "D", "KK", "a" ]
```

3.7 Karşılaştırma Operatörleri

< , > , <= , >= , <> , = operatörleri, GAP programında kullanılan temel karşılaştırma operatörleridir [17].

- < operatörü aynı aileden iki değerin küçüklük karşılaştırmasını gerçekleştirmek için ,
- > operatörü aynı aileden iki değerin büyüklük karşılaştırmasını gerçekleştirmek için,
- <= operatörü aynı aileden iki değerin küçük eşitlik karşılaştırmasını gerçekleştirmek için,
- >= operatörü aynı aileden iki değerin büyük eşitlik karşılaştırmasını gerçekleştirmek için,
- = operatörü aynı aileden iki değerin eşitlik karşılaştırmasını gerçekleştirmek için,

- <> operatörü aynı aileden iki değerin eşit değil mi karşılaştırmasını gerçekleştirmek için kullanılır.

```
gap> a:=2;
2
gap> b:=3;
3
gap> a<b;
true
gap> a>b;
false
gap> a=b;
false
gap> a<>b;
true
```

Karşılaştırma operatörleri, geçişmeli yani parantezlerden bağımsız değildir. GAP programında karşılaştırma operatörleri kullanılırken uygun şekilde parantezleme işlemi yapılmalıdır.

3.8 Aritmetik Operatörler

GAP programında kullanılan temel aritmetik operatörler aşağıda verilmiştir [17].

```
+     -     *     /     ^     mod
```

- a+b komutu, a ve b sayılarının toplamı,
- a-b komutu, a sayısından b sayısının çıkartılması,
- a*b komutu, a ve b sayılarının çarpımı,
- a/b komutu, a sayısının b sayısına bölünmesi,
- a^b komutu a sayısının b. dereceden kuvvetinin alınması,
- a mod b komutu, a sayısının mod b cinsinden değerinin bulunması anlamına gelir. - operatörü bir sayının işaretini değiştirmek için de kullanılabilir.

```
gap> x:=10;
10
```

```
gap> y:=5;
5
gap> z:=13;
13
gap> x*z;
130
gap> x/y;
2
gap> y^z;
1220703125
gap> z mod y;
3
gap> -z;
-13
```

Matematiksel işlemlerde olduğu gibi GAP programında da işlemlerin bir öncelik sırası vardır. () işaretleri arasında yazılan işlemler en yüksek işlem önceliğine sahiptir. () işaretlerinden sonra en yüksek işlem önceliği ^ (üst alma) aritmetik operatörüne aittir. Aşağıdaki tablo, GAP programında aritmetik operatörlerin işlem önceliğini göstermektedir. Tabloda en üst satırdaki aritmetik operatör en yüksek önceliği gösterir.

```
^
* /
+ -
```

GAP programında aynı satırda bulunan aritmetik operatörlerden daha solda bulunan operatör daha yüksek işlem önceliğine sahiptir.

```
gap> a:=1-(2-3);
2
gap> b:=(1-2)-3;
-4
gap> a=b;
false
gap> 1+2^3;
9
```

```
gap> (1+2)^3;
27
gap> (-(2^(-2))*3)+1;
1/4
```

3.9 Mantıksal Operatörler

GAP programı kullanılırken, kıyaslaması yapılan değerler arasında mantıksal operatörler kullanılabilir. Bu mantıksal operatörler, `not` (değil) , `or` (veya) ve `and` (ve) den ibarettir. Buradaki `or` (veya) mantıksal operatöründe kıyaslaması yapılan iki değerden birisi doğru iken doğru , `and` (ve) mantıksal operatöründe ise kıyaslaması yapılan iki değer de doğru iken doğrudur [17].

```
gap> not true ;
false
gap> true and false ;
false
gap> true or false;
true
gap> 12 >= 0 and 2 <> 2;
false
gap> 12 >= 0 or 2 <> 2;
true
```

3.10 Program Denetim Deyimleri

3.10.1 If

Koşullu işlemler gerçekleştirmek için kullanılan bir deyimdir. `If` denetim deyiminin genel kullanım şekli aşağıdaki gibidir [17].

```
if kosul1 then
deyim1;
elif kosul2 then
deyim2;
else
deyim3;
```

```
fi;
```

Koşulların yorumlanmasına en üstte yer alan `if` deyiminden başlanır. Buradaki `kosul1` olumlu ise `deyim1` de yer alan ifadeler gerçekleştirilir ve program akışı `fi` komutundan itibaren devam eder. Aksi halde, yani `kosul1` olumlu değil ise, program `elif` komutunun bulunduğu satıra gelir ve `kosul2` bölümünde yer alan kıyaslamanın doğruluğunu test eder. `kosul2` olumlu ise `deyim2` de yer alan ifadeler gerçekleştirilir ve program akışı `fi` komutundan itibaren devam eder. `if` ve `elif` komutlarındaki `kosul1` ve `kosul2` bölümlerinde yer alan kıyaslamalar doğru değil ise, program `else` komutunun bulunduğu satıra gelir ve `deyim3` de yer alan ifadeler gerçekleştirilir. `if` deyimi kullanılırken `elif` ve `else` komutlarını kullanmak zorunluluğu yoktur. `elif` komutu bir `if` deyimi içersinde istenilen çoklukta kullanılabilir.

```
gap> k:=-52;
-52
gap> if 0<k then
> sgnk:=1;
> elif k<0 then
> sgnk:=-1;
> else
> sgnk:=0;
> fi;
gap> sgnk;
-1
```

3.10.2 While

Belirli bir koşulla birlikte bir döngü oluşturmak için kullanılan bir deyimdir. `while` denetim deyiminin genel kullanım şekli aşağıdaki gibidir [17].

```
while kosul1 do
deyim1;
od;
```

Koşulların yorumlanmasına, en üstte yer alan `while` deyiminden başlanır. Buradaki `kosul1` olumlu ise `deyim1` de yer alan ifadeler gerçekleştirilir. Daha sonra program `od;`

komutuna geldiğinde tekrar `while` deyiminin olduğu satıra gelir ve `kosul1` in doğruluğunu test eder. `kosul1` olumsuz oluncaya kadar, yani `kosul1` deki kıyaslama yanlış oluncaya kadar bu döngü devam eder. `kosul1` olumsuz olduğunda program `od;` satırından sonraki ilk satırdan işlemlerine devam eder.

```
gap> para:=10;
10
gap> yil:=0;
0
gap> while para<50 do
> para:=para+(para*(30/100));
> yil:=yil+1;
> od;
gap> yil;
7
gap> para;
62748517/1000000
```

Yukarıdaki program, 10 liranın yıllık %30, faizle 50 lirayı geçtiği yada eşit olduğu ilk yılı ve paranın kaç lira olduğunu hesaplamak için kullanılabilir. `para<50` koşulu sağlanıncaya kadar `while` deyimi tekrarlanmaya devam edecektir. `para>=50` değeri sağlandıktan sonra koşul sağlanamayacak ve döngüden çıkılacaktır. GAP programında `while` deyimi kullanılırken dikkat edilmesi gereken önemli bir konu, koşulda sınanan değerin döngü içersinde değiştirilme gerektiğidir. Aksi halde sonsuz döngüye girilir ki bu GAP programının yanıt veremez hale gelmesi demektir. Yukarıdaki örnekte `while` içerisinde `para` değişkeninin değeri değiştirilmeseydi `para<50` koşulu sürekli olarak sağlanacağından program sonsuz döngüye girerdi.

3.10.3 Repeat

Belirli bir koşulla birlikte bir döngü oluşturmak için kullanılan başka bir deyimdir. `repeat` denetim deyiminin genel kullanım şekli aşağıdaki gibidir [17].

```
repeat
deyim1;
until kosul1;
```

repeat denetim deyiminde hiçbir koşula bakılmaksızın deyim1 de yer alan ifadeler gerçekleştirilir. Daha sonra until komutunun bulunduğu satırda yer alan kosul1 test edilir. Buradaki kosul1 sağlanıncaya kadar program repeat satırının olduğu yere gidecek ve deyim1 de yer alan ifadeler tekrar gerçekleştirilecektir. kosul1 sağlandığında program until komutunun bulunduğu satırdan bir sonraki satıra geçerek işlemlerine devam edecektir.

repeat denetim deyimi ile while denetim deyimi yapı ve kullanım şekli olarak birbirine çok benzemektedir. repeat ve while arasındaki belirgin fark, while deyiminde işlemler yapılmadan önce koşulun sağlanması gerektiği, repeat deyiminde ise koşula bakılmaksızın işlemlerin gerçekleştirip daha sonra koşula bakılmasıdır. repeat ve while deyimleri arasındaki önemli bir diğer fark ise, while ile oluşturulan döngünden koşul olumsuz olduğunda, repeat ile oluşturulan döngüden ise koşul sağlandığında çıkılır.

```
gap> para:=10;
10
gap> yil:=0;
0
gap> repeat
> para:=para+(para*(30/100));
> yil:=yil+1;
> until para>=50;
gap> yil;
7
gap> para;
62748517/1000000
```

Yukarıdaki program, repeat deyimi ile 10 liranın yıllık %30 faizle 50 lirayı geçtiği yada eşit olduğu ilk yılı ve paranın kaç lira olduğunu hesaplamak için kullanılabilir. para>=50 koşulu sağlanıncaya kadar repeat deyimi tekrarlanmaya devam eder. para>=50 değeri sağlandıktan sonra koşul sağlanacak ve döngüden çıkılacaktır. While deyiminde olduğu gibi repeat deyimi kullanılırken de koşul içersindeki değer döngü içersinde değiştirilmez ise program sonsuz döngüye girebilir. Yukarıdaki örnekte repeat içerisinde para değişkeninin değeri değiştirilmeseydi para>=50 koşulu hiçbir zaman sağlanamayacak ve program sonsuz döngüye girecekti.

```
gap> a:=2;
2
gap> b:=4;
4
gap> repeat
> b:=b+1;
> a:=b^2;
> until a>0;
gap> a;
25
gap> b;
5
```

Yukarıdaki örnekte, a>0 durumu sağlandığı halde, döngü bir kez çalışmış a ve b değişkenlerinin tuttuğu değerler değişmiştir.

3.10.4 For

Belirli bir koşulla birlikte bir döngü oluşturmak için kullanılan başka bir deyimdir. for denetim deyiminin en önemli özelliği, bir döngü sayacına sahip olmasıdır. for denetim deyiminin genel kullanım şekli aşağıdaki gibidir [17].

```
for i in [1..n] do
deyim1;
> od;
```

for denetim deyiminde hiçbir koşula bakılmaksızın deyim1 de yer alan ifadeler gerçekleştirilir. Burada verilen [1..n] gösterimi, ilk elemanı 1 olan ve birer artışla son elamanı n ye ulaşan bir listeyi ifade etmektedir. [..] ifadesi, ilk elemandan son elemana doğru birer birer artışı göstermektedir. i değişkenin ilk değeri verilen listenin ilk elemanıdır. Program deyim1 de yer alan ifadeleri gerçekleştirip od; komutunun bulunduğu satıra geldiğinde tekrar for deyiminin olduğu satıra gelir ve i değeri verilen listenin ikinci elemanın değerini alır. Döngü, verilen liste elemanlarının sayısınca çalışacaktır. Döngü, son kez çalıştığında yani i değeri, verilen listenin son elemanın değerini alarak od; komutunun bulunduğu satıra geldiğinde herhangi bir koşul sınaması yapmadan bu satırdan sonraki satırdan işleme devam edecektir.

```
gap> faktoriyel:=1;
1
gap> for i in [1..6] do
> faktoriyel:=faktoriyel*i;
> od;
gap> faktoriyel;
720
```

Yukarıdaki örnekte `for` deyimi ile oluşturulan döngü altı kez çalışacaktır. Örnekte `i` değeri `[1..6]` listesinin tüm elemanlarının değerini bir kez alacaktır. (Döngü 1. kez dönerken `i=1` , 2. kez dönerken `i=2`, 6. kez dönerken `i=6` gibi.) `faktoriyel` değişkeni ile birlikte döngü altı kez çalıştıktan sonra $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$ değeri hesaplanacaktır.

GAP programında kullanılacak listelere bir tanımlayıcı ismi verilebilmektedir. GAP programında liste kullanmak konusunda sonraki bölümlerde ayrıntılı bilgi verilecektir. Burada `for` denetim deyimi açıklanırken kullanılacak bir listenin uzunluğunun `Length(liste)` komutu ile belirlendiğini belirtmek gerekir. `For` denetim deyimi ile birlikte tanımlayıcı ismi verilen listeler de kullanabilir.

```
gap> liste:=[1,3,5,7,9,15];
[ 1, 3, 5, 7, 9, 15 ]
gap> for k in liste do
> Print(k, " ");
> od; Print("\n");
1 3 5 7 9 15
```

Yukarıdaki örnekte `liste` tanımlayıcı ismi ile oluşturulan liste elemanları sırası ile `k` değerini almışlardır. Sonraki bölümde ayrıntılı olarak bahsedilecek olan `Print()` komutunun şu an için bir tanımlayıcı isminin tuttuğu değeri ekrana yazdırdığını bilmek yeterlidir.

`for` denetim deyimi, listeler dışında kümeler veya gruplarla birlikte de kullanılabilir. GAP programında grup cebirsel yapılarından 9. bölümde bahsedilecektir. `Order()` komutu grubun bir elemanının mertebesini bulmak için kullanılmıştır.

```
G:=Group((1,2,3,4,5),(6,7,8,9,10),(1,2));
Group([ (1,2,3,4,5), (6,7,8,9,10), (1,2) ])
gap> eleman:=0;
0
gap> elmertop:=0;
0
gap> for i in G do
> eleman:=eleman+1;
> elmertop:=elmertop+Order(i);
> od;
gap> eleman;
600
gap> elmertop;
7971
```

Yukarıdaki örnekte bir G grubu oluşturulmuştur. Bu G grubunun `for` denetim deyimi içerisinde kullanılması, `i` değişkeninin grubun birinci elemanından başlayarak son elemana kadar değerler alması anlamına gelir. Döngü sona erdiğinde `eleman` değişkeni G grubunun eleman sayısını (yani G grubunun mertebesini) ve `elmertop` değişkeni, G grubunun elemanlarının mertebelerinin toplamını bulmuş olacaktır.

`for` denetim deyiminin hiçbir koşula bakılmaksızın liste elemanlarının sayısınca çalıştırıldığından bahsedilmişti. GAP programı kullanılırken `i` değerinin birer birer artması yerine farklı adımlarla artması veya azalması istenilen programlar yazılması gerekebilir. `while` denetim deyimi ihtiyaç halinde `for` denetim deyimine benzer şekilde kullanılabilir.

```
dongu_listesi := liste;
i:= 1;
while i<= Length(dongu_listesi) do
deyim1;
i:=i+1;
od;
```

GAP programı kullanılırken sonsuz döngüye düşmemek için `i` değişkeninin döngü içerisinde değiştirilmesi gerektiği unutulmamalıdır.

```
gap> Liste:=[1,2,3,4,5,6,7,8,9,10,12,14,15];
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15 ]
gap> i:=2;
2
gap> while i<=Length(Liste) do
> Print(i," ");
> i:=i+3;
> od; Print("\n");
2 5 8 11
```

Program denetim deyimleri, birbirleri içerisinde kullanılarak iç içe döngüler oluşturulabilir.

3.10.5 Break

Denetim deyimi ile oluşturulmuş bir döngüden çıkmak için kullanılan deyimdir. Döngü içersinde `break` deyimi ile karşılaşıldığı zaman hiçbir koşul dikkate alınmadan döngü sonlanır ve program akışı döngüden sonraki ilk satırdan itibaren devam eder. İç içe döngüler içersinde `break` deyimi kullanıldığında ise en içteki döngüden çıkılır [17].

```
gap> G:=Group((1,2,3,4,5),(8,9));
Group([ (1,2,3,4,5), (8,9) ])
gap> for i in G do
> if Order(i)=2 then
> break;
> fi;
> od;
gap> i;
(8,9)
gap> Order(i);
2
gap>
```

Yukarıdaki örnekte bir G grubu oluşturulmuş ve `if` deyimi ile G grubunun elemanlarının mertebelerinin 2 ye eşit olup olmadığı koşulu incelenmiştir. G grubunda, mertebesi 2 ye eşit olan ilk elemana sıra geldiğinde `break` deyimi işlenecek ve döngüden

çıkılacaktır. Döngüden çıkıldığı anda i değişkeni, mertebesi 2 olan ilk elemanı tutuyor olacaktır.

Genellikle if deyimi ile birlikte kullanılan break deyiminin, mutlaka bir döngü içerisinde kullanılması gerekir.

3.10.6 Continue

Denetim deyimi ile oluşturulmuş bir döngüden, içerisinde belli özellikteki çevrim işlemleri gerçekleştirmeden sonraki çevrime geçmek için kullanılan deyimdir. Döngü içersinde continue deyimi ile karşılaşıldığı zaman ondan sonra gelen deyimler atlanır ve döngü bir sonraki çevrime girer. Bir bakıma break deyiminin tersi gibi düşünülebilir. break döngüyü sonlandırma görevini continue ise bir sonraki çevrime geçirme görevini yürütür [17].

```
gap> G:=Group((1,3,2),(1,3));
Group([ (1,3,2), (1,3) ])
gap> for k in G do
> if Order(k)=4 then
> continue;
> fi;
> Print(k,"\n");
> od;
()
(1,2,3)
(1,3,2)
(2,3)
(1,2)
(1,3)
gap> Order(());
1
gap> Order((1,2,3));
3
gap> Order((1,3,2));
3
gap> Order((2,3));
2
gap> Order((1,2));
```

```
2
gap> Order((1,3));
2
```

Yukarıdaki örnekte bir G grubu oluşturulmuş ve `if` deyimi ile G grubunun elemanlarının mertebelerinin 4 e eşit olup olmadığı koşulu incelenmiştir. G grubunda mertebesi 4 e eşit olan ilk elemana sıra geldiğinde `continue` deyimi işlenecek ve döngü içerisinde yer alan `Print()` komutu, G nin mertebesi 4 olan elemanı için işletilmeyecek ve döngü bir sonraki eleman için devam edecektir. Döngüden çıkıldığında G grubunun mertebeleri 4 den farklı olan elemanları yazdırılmış olacaktır.

`continue` deyimi de `break` deyiminde olduğu gibi genellikle `if` deyimi ile birlikte ve mutlaka bir döngü içerisinde kullanılmalıdır.

3.10.7 Print

Bu komut, bir veya daha fazla tanımlayıcı isminin tuttuğu değeri ekrana yazdırmak için kullanılır. `Print` komutu genel olarak döngüler içerisinde kullanılır. Birden fazla tanımlayıcı ismi tek bir komut içerisinde virgülle birbirinden ayrılarak kullanılabilir [17].

```
gap> A:=[1,2,3];
 [ 1, 2, 3 ]
gap> Print(A);
[ 1, 2, 3 ]gap> B:=[1,2,3,4];
[ 1, 2, 3, 4 ]
gap> Print(B);
[ 1, 2, 3, 4 ]gap>
```

Yukarıdaki örnekte görüldüğü gibi `Print` komutu yalın olarak kullanıldığında `gap>` ifadesi bir alt satırdan değil de yazdırılan tanımlayıcı ismi değerinden sonra gelecektir. GAP programında bu gibi durumlar için kullanılabilecek özel parametreler vardır.

`\n`

Bu karakter GAP programında, "" işaretleri arasında, imlecin bir satır aşağıya atılması için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır.

```
 gap> Print(A,B);
[ 1, 2, 3 ][ 1, 2, 3, 4 ]gap> Print(A,"\n",B,"\n");
[ 1, 2, 3 ]
[ 1, 2, 3, 4 ]
```

\"

Bu karakter GAP programında, "" işaretleri arasında, ekrana " işaretinin çıkarılması için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır.

```
gap> Print("\"",A,"\"",B,"\"","\n");
"[ 1, 2, 3 ]"[ 1, 2, 3, 4 ]"
```

\'

Bu karakter GAP programında, "" işaretleri arasında, ekrana ' işaretinin çıkarılmasını için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır.

```
gap> Print("\'",A,"\'",B,"\'","\n");
'[ 1, 2, 3 ]'[ 1, 2, 3, 4 ]'
```

\\

Bu karakter GAP programında, "" işaretleri arasında, ekrana \ işaretinin çıkarılması için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır.

```
gap> Print("\\",A,"\\",B,"\\","\n");
\[ 1, 2, 3 ]\[ 1, 2, 3, 4 ]\
```

\b

Bu karakter GAP programında, "" işaretleri arasında, ekrana yazdırılan ifadeye backspace (geriye boşluk) işlemini gerçekleştirmek için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır. Bu karakter kullanıldığı yerde kendisinden önceki ilk karakteri siler.

```
gap> Print(A,"\b",B,"\n");
[ 1, 2, 3 [ 1, 2, 3, 4 ]
```

GAP programında `Print` komutunu kullanarak tanımlayıcı isminin, tuttuğu değerden başka yazılmak istenilen herhangi bir açıklama da ekrana yazdırılabilir. Açıklama GAP programında, "" işaretleri arasında kullanılır. Açıklamalar bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır. Tanımlayıcı isimlerinin tuttuğu değerler arasına boşluk bırakmak için de içerisi boş "" işaretleri kullanılabilir.

```
gap> Print(A,"\n Yukarida A degerini yazdirdik\n");
[ 1, 2, 3 ]
 Yukarida A degerini yazdirdik
gap> Print("   ",A,"   ","\n Yukarida A degerini yazdirdik\n");
   [ 1, 2, 3 ]
   Yukarida A degerini yazdirdik
```

\t

Bu karakter GAP programında, "" işaretleri arasında, kendisinden önce ve sonra gelen karakterler arasında bir tab (sekiz kez boşluk tuşuna basmak) boşluk bırakmak için kullanılır. Bu karakter bir tanımlayıcı ismi ile birlikte kullanılıyorsa araya virgül koyulmalıdır.

```
gap> Print("\t",A,"\n","        ",A,"\n");
       [ 1, 2, 3 ]
       [ 1, 2, 3 ]
```

GAP programında `Print` komutu genel olarak döngü içerisinde kullanılır. Bu komut içerisinde ki değişkenlere temel aritmetik işlemler de yaptırılabilir.

```
gap> liste:=[1,2,3,4,5,6,7];
[ 1, 2, 3, 4, 5, 6, 7 ]
gap> for i in liste do
> Print(i," ",2*i," ",i^2,"\n");
> od;
1 2 1
2 4 4
3 6 9
4 8 16
5 10 25
6 12 36
7 14 49
```

4 GAP PROGRAMI KULLANMAYA BAŞLAMAK

4.1 Giriş

Bu bölümde GAP programının yeni kullanıcıları için programın kullanılması basit örneklerle açıklanmıştır. Bu açıklamalar yapılırken GAP programının kullandığı en küçük bir işaret dahi örneklerle sunulmuştur. Bunun için sık sık kullanılan GAP programı komutları ki bunlar `quit ,` `LogTo()` komutları ve Crtl tuşuyla yapılan basit kombinasyon komutları bu bölümde verilmiştir. Bu bölümde GAP programının kullanımında `gap>` promptu var iken yapılabilecek komut kullanımları örneklere sunulmuştur. Bu bölümde sunulan en önemli komutlar `help , LogTo()` ve `last` komutlarıdır.

Bu bölümde GAP programında sıkca kullanılan liste yöntemlerini ve bu yöntemlerin kullanıldığı temel fonksiyonları basit örnekleriyle birlikte listeler başlığı altında sunulmuştur. GAP programında program yapımına temel teşkil eden fonksiyon kullanım yöntemleri ve bu yöntemlerin kullanım örnekleri de bu bölüm içerisinde sunulmuştur.

4.2 Sık Sık Kullanılan Bazı GAP Programı Komutları

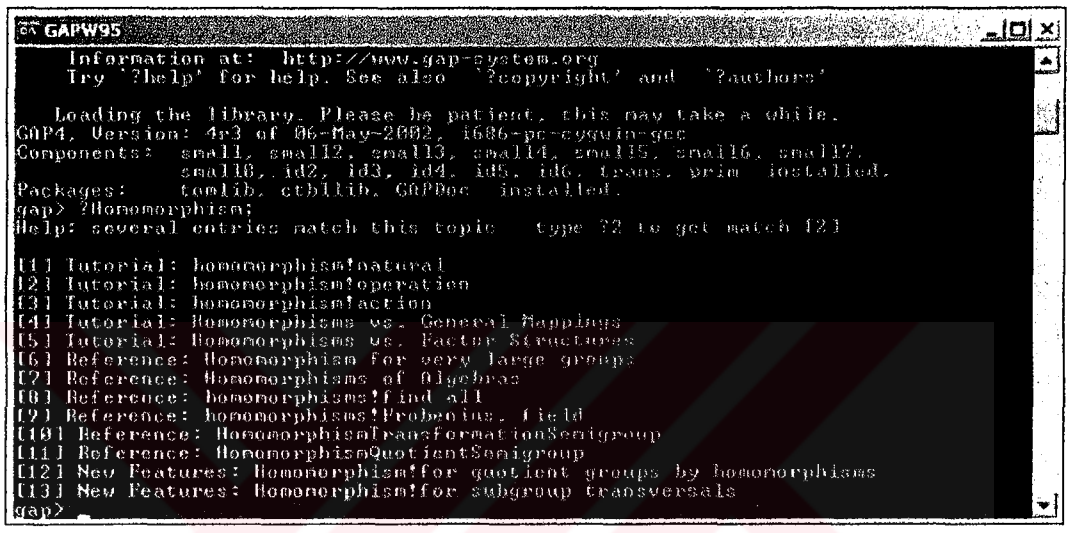
- GAP programında bazı istisnalar dışında her komut ; işareti ile bitmelidir [16].
- GAP programından çıkmak için `quit;` komutu kullanılır. Ctrl+D tuşlarına birlikte basarak ta GAP programından çıkılabilir.
- Yapılan çalışmaların kaydedilmesi için `LogTo("DosyaAdı");` komutu kullanılır. Çalışma sayfasının kaydedilmesinin sonlandırılması için `LogTo();` komutu kullanılır.
- Ctrl+P tuşlarına birlikte basılırsa bir önceki komut tekrar görüntülenir.
- Ctrl+E tuşlarına birlikte basıldığında imleç bulunduğu satırın en sonuna gider.
- Ctrl+B tuşlarına birlikte basıldığında imleç bulunduğu satırda bir önceki karakterin altında yer alır.
- Eğer GAP programı bir hata döngüsüne girer ve prompt `brk>` halini alırsa, bu döngüden çıkıp promptu normal hale yani `gap>` haline getirebilmek için Ctrl+D tuşuna basmak veya `quit;` komutunu vermek yeterlidir.

4.3 GAP Programı Kullanımı

GAP programı kullanırken herhangi bir komut hakkında bilgi almak için `?komut;` şeklinde komut verilmesi gerekir. Hakkında bilgi sahibi olmak istenilen komut tek şekilde

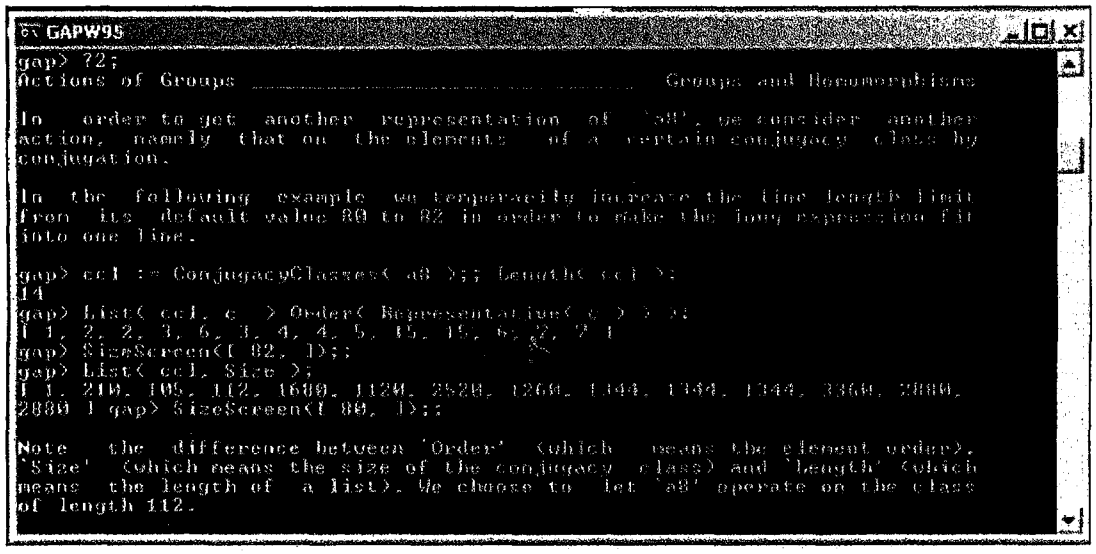
kullanılıyorsa GAP programı doğrudan yardım dosyasını açacaktır. Fakat yazılan komut birden fazla şekilde kullanılabiliyorsa `?komut;` komutu verildiğinde bu komut hakkında yardım alınabilecek konu başlıkları bir numaraya göre listelenir.

GAP programında yardım özelliği bir örnekle aşağıdaki şekilde gösterilebilir. `?Homomorphism;` komutu ile homomorfizmler hakkında yardım istenir.



Şekil 4.1 GAP programı yardım ekranı

Yukarıdaki ekranda içerisinde homomorfizm kelimesi geçen 13 tane farklı konu hakkında yardım elde edilmiş olur. Bu konular arasında seçim yapmak için `?numara` komutu kullanılır (`[2]` konusu için `?2` gibi). Bu komut verildikten sonra o konu hakkındaki yardım dosyası ekranda görüntülenir.



Şekil 4.2 GAP programı yardım konusu seçilmesi

Yardım dosyasında verilen bilgiler GAP programı penceresine sığmıyorsa sayfalar ve satırlar arasında geçiş yapmak için aşağıdaki klavye tuşları kullanılabilir.

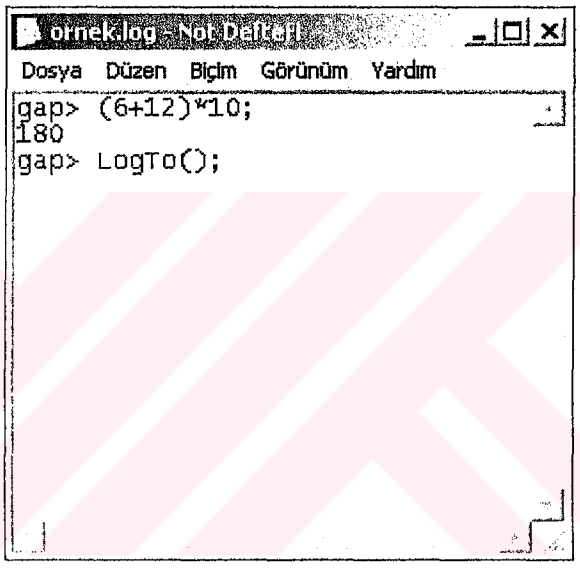
- **n** : Bir sayfa dolduktan sonra bir sonraki satırı görmek için bu tuş kullanılır.
- **p** : Bir sayfa dolduktan sonra birkaç satır geriye gelinip tekrar önceki satırlara girmek istenirse kullanılır.
- **boşluk(space)** : Bir sayfa dolduktan sonra bir sonraki sayfayı görmek için bu tuş kullanılır.
- **b** : Bir sayfa dolduktan sonra diğer sayfaya geçilmişse tekrar önceki sayfayı görmek için bu tuş kullanılır.
- **q** : Yardım sayfalarından çıkmak için bu tuş kullanılır.

GAP programı kullanırken büyük ve küçük harf kullanmak arasında fark vardır. Örneğin `LogTo` ve `Logto` aynı işlevi yapmazlar.

GAP programı kullanımının daha iyi anlaşılması için aşağıdaki örnekşer verilebilir. bazı basit örneklerle çalışılmıştır. Öncelikle `LogTo` ile program çıktılarının kaydedileceği dosyaların gürülebilmesini sağlamak için `LogTo("ornek.log");` komutu kullanarak, ornek isimli ve log uzantılı metin dosyası masaüstünde yer alması sağlanır. Bu komut GAP programı kullanırken yazdığımız her komutu ve aldığımız sonuçları `ornek.log` metin dosyasına kaydeder. Bu kayıt işlemi `LogTo();` komutu kullanılarak sonlandırılır.

```
gap> LogTo("ornek.log");
gap> (6+12)*10;
180
LogTo();
```

Masaüstüne kaydedilen `ornek.log` adlı dosya herhangi bir metin editörü ile açılabilir. Windows ortamında herhangi bir farklı ayarlama yapılmadığı durumlarda `log` uzantılı dosyalar NotDefteri programı kullanılarak açılır.



Şekil 4.3 Ornek.log dosyasının görünümü

GAP programının `gap>` promptunda aynı `(6+12)*10` matematiksel hesaplamasını noktalı virgül olmadan yazıp enter tuşuna basılması durumunda;

```
gap> (6+12)*10
>
```

çıktısı elde edilecektir. Bu GAP programının hiçbir yanıt vermemesi demektir. GAP programı sonunda noktalı virgül kullanılmadan yazılan komutlar için bir hata vermez ancak o komutun bittiğini anlayamayacağından bir sonuç vermez. Bu tip işlemlerde sonuç alınmak isteniyorsa noktalı virgül bir sonraki satırda da kullanılabilir.

```
gap> (6+12)*10
> ;
180
```

GAP programı kullanılırken iki komut aynı satırda birlikte yazılabilir. Bunun için her komuttan sonra bir tane noktalı virgül kullanılması gerekir. Bununla birlikte bir komutun sonunda iki tane noktalı virgül birlikte de kullanılabilir. Bunun anlamı o komutun sonucunun ekrana yazdırılmaması demektir [16].

```
gap> -3; 17-25;
-3
-8
gap> -3;; 17-25;
-8
gap> -3; 17-25;;
-3
gap> -3;; 17-25;;
gap>
```

GAP programında matematiksel işlemler yaparken işlem önceliğine de dikkat edilmelidir. Örnekteki () işaretleri kullanılmazsa bulunacak değer daha farklı olacaktır.

```
gap> 6+12*10;
126
```

GAP programı kullanırken bir komutta açılan her parantez uygun bir biçimde kapatılmalıdır. Aksi durumda program hata verir.

```
gap> (6+12*10;
Syntax error: ) expected
(6+12*10;
        ^
```

Yukarıdaki gibi bir hata mesajı ile karşılaştığında Ctrl+P tuşlarına birlikte basılarak bir önceki komuta tekrar dönülebilir. Daha sonra hataya neden olan parantez uygun şekilde yerleştirilerek tekrar enter tuşuna basılmalıdır.

```
gap> (6+12*10;
Syntax error: ) expected
(6+12*10;
        ^
gap> (6+12)*10;
180
```

GAP programı değişkenlere bir isim atanmasına da izin vermektedir. Değişkenlere değer atanması := ifadesi ile yapılır. GAP programında değişken isimlerine identifier (tanımlayıcı) denilmektedir. Aşağıdaki örneklerde a ve b birer identifier (tanımlayıcı) lardır.

```
gap> a:=(11+7)*(11-7);
72
gap> a;
72
gap> b:=a*(a-1);
5112
gap> b;
5112
gap> a:=20;
20
gap> b:=a*(a-1);
380
gap> b;
380
gap> a=b;
false
```

GAP programı matematiksel değerler dışında karakterlere de atama işlemi yapabilir. Karakterlere atama işlemi yapılırken ataması yapılacak karakter '' işaretleri arasında belirtilmelidir.

```
gap> a:='D';
'D'
gap> b:='P';
'P'
```

```
gap> c:='U';
'U'
gap> a; b; c;
'D'
'P'
'U'
```

Identifier (tanımlayıcı) olarak kullanılan bir isme atanılan değer daha sonraki satırlarda değiştirilmediği sürece, devamlı olarak değeri sabit kalacaktır. Identifier (tanımlayıcı) olarak atanan isimlerde harfler ve rakamlar kullanılabilir ancak dikkat edilmesi gereken hassas bir nokta da her identifier (tanımlayıcı) isminin en azından bir tane harf içermesidir.

GAP programı kullanılırken bazen bir önceki sonuçla ilgili işlem yapma gereksinimi de duyulabilir. Bunun için `last` komutunun kullanılır. Bu komut bir satır önce yapılan işlemin sonucunu verecektir [16].

```
gap> (12-15)*(2^4);
-48
gap> a:=last;
-48
```

GAP programında ^ parametresi bir sayının kuvvetini (üssünü) almak için kullanılır. GAP programı çok büyük sayıları ekrana yazmaz fakat bu sayıları hesaplar ve işlemlerde kullanmasına izin verir. GAP programında `Factorial()` komutu kullanılarak faktöriyel hesaplaması yapılabilir [17].

```
gap> 2^4;
16
gap> 2^56;
72057594037927936
gap> a:=10^10000;
<<an integer too large to be printed>>
gap> b:=10^10;
10000000000
gap> a/(b^1000);
1
```

```
gap> Factorial(4);
24
gap> Factorial(5);
120
gap> Factorial(15);
1307674368000
```

GAP programı iki değerin eşitliğini kontrol edebilir. Eşitliği sorulan iki değer için GAP programı `true` veya `false` yanıtı verecektir.

```
gap> 6=7;
false
gap> (9-6)*15=9-6*15;
false
```

GAP programı eşitliklerin kıyaslamasından daha fazla olarak `<>` , `<` , `<=` , `>` ve `>=` operatörleri yardımıyla eşit değil mi? , küçük mü? , küçük eşit mi? , büyük mü? ve büyük eşit mi? kıyaslamalarını da yapabilir [17].

```
gap> 2^5 < 2^2;
false
gap> 2^5 <> 2^2;
true
```

GAP programında `IsPrime()` komutu bir sayının asal sayı olup olmadığını test etmek için kullanılır [17].

```
gap> IsPrime(2);
true
gap> IsPrime(4);
false
gap> IsPrime(571);
true
```

GAP programı kullanıken yapılan bir işlemin ne kadar süre aldığı time komutu kullanılarak bulunabilir. Bu komut yapılan işlem için geçen süreyi milisaniye cinsinden verecektir [17].

```
gap> G:=Group((1,2,3,4),(1,2,3));
Group([ (1,2,3,4), (1,2,3) ])
gap> time;
62
gap> Order(G);
24
gap> time;
79
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> time;
15
```

4.4 Listeler

Listeler köşeli parantezler içerisinde virgülle ayrılmış nesneler topluluğudur. Listeler oluşturulurken rakamlardan farklı olarak harfler veya isimlerde kullanılabilir bu durumda bu elemanlar "" işaretleri arasında yazılmalıdır. Aynı liste içerisinde hem harfler hem de rakamlar kullanılabilir. Şimdi GAP programı kullanarak asal sayıların ilk yedi elemanıyla bir liste oluşturalım

```
gap> dpu:=["d","p","u"];
[ "d", "p", "u" ]
gap> dpu2:=["D","P","U",4,3];
[ "D", "P", "U", 4, 3 ]
gap> asal:=[2,3,5,7,11,13,17];
[ 2, 3, 5, 7, 11, 13, 17 ]
```

Oluşturulmuş olan bir listeye yeni elemanlar eklemek için Append() komutu kullanılır. Append() komutu ile eklenecek elemanlar yine bir liste biçiminde verilmelidir. Örneğin yukarıda oluşturulan asal adlı listeye asal sayıların 8-11. elemanları da eklenirse asal adlı listenin yeni hali aşağıdaki gibi olacaktır [17].

```
gap> Append (asal,[19,23,29,31]);
gap> asal;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31 ]
```

Add() komutu da yine bir listeye eleman eklemek için kullanılır. Add() komutu Append() komutundan farklı olarak listeye yeni bir liste değil, sadece yeni bir eleman eklemek için kullanılır. Asal adlı listeye asal sayıların 12. elemanını eklenirse listenin yeni hali aşağıdaki gibi olacaktır [17].

```
gap> Add(asal,37);
gap> asal;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 ]
```

Length() komutu oluşturulmuş olan bir listenin uzunluğunu, yani kaç elemanlı olduğunu gösterir. Liste_ismi[eleman_sırası] komutu kullanılarak listenin kaçıncı elemanının ne olduğu anlaşılabilir. Listeye eklemek istenilen herhangi bir elamanın listenin kaçıncı elemanı olacağı belirtilerek liste içine alınabilir. Örnek olarak asal sayılar listesine 13. ve 14. eleman olarak 41 ve 43 sayıları eklenirse listenin yeni hali aşağıdaki gibi olacaktır.

```
gap> Length(asal);
12
gap> asal[8];
19
gap> asal[1];
2
gap> asal[13]:=41;
41
gap> asal[14]:=43;
43
gap> asal;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43 ]
```

Asal adlı liste şu an 14 elemandan oluşmaktadır. Asal sayıların 20. elemanın 71 olduğu akla gelir ve bu listeye eklenmek istenirse aşağıdaki komutu kullanmak yeterlidir. Böylece listenin uzunluğu 20 olacaktır. Eğer GAP programına listenin 15. elemanının ne olduğu sorulursa program hata verecektir çünkü 15. eleman tanımlı değildir [17].

```
gap> asal[20]:=71;
71
gap> asal;
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,,,,, 71 ]
gap> Length(asal);
20
gap> asal[15];
List Element: <list>[15] must have an assigned value
not in any function
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' after assigning a value to continue
brk>
```

GAP programında liste özellikleri kullanılarak değer atamaları yapılan değişkenlere bir liste oluşturularak bu değişkenlerle dört işlem aşağıdaki gibi gerçekleştirilebilir. İşlemler gerçekleştirilirken listenin kaçıncı elemanı olduğunu belirtilebileceği gibi doğrudan değişken isimleri de yazılabilir.

```
gap> a:=15;
15
gap> b:=20;
20
gap> c:=25;
25
gap> d:=30;
30
gap> e:=35;
35
gap> liste:=[a,c,b,e,d];
[ 15, 25, 20, 35, 30 ]
```

```
gap> carpma:=liste[1]*liste[3];
300
gap> bolme:=d/a;
2
gap> toplama:=liste[2]+b;
45
gap> cikarma:=a-liste[4];
-20
```

Listeler birbiri içerisinde kullanılabilir ve her bir liste içindeki elemanlar bu listeleri kapsayan liste içerisinden çağrılabilir.

```
gap> a:=[10,12,14,16,18];
[ 10, 12, 14, 16, 18 ]
gap> b:=[11,13,15,17,19];
[ 11, 13, 15, 17, 19 ]
gap> c:=[10,11,12,13,14];
[ 10, 11, 12, 13, 14 ]
gap> d:=[10,15,20,25,30];
[ 10, 15, 20, 25, 30 ]
gap> e:=[10,20,30,40,50];
[ 10, 20, 30, 40, 50 ]
gap> liste:=[a,c,b,d,e];
[[10,12,14,16,18],[10,11,12,13,14],[11,13,15,17,19 ],
[10,15,20,25,30],[10,20,30,40,50]]
gap> liste[3][2]=b[2]; liste[1][1]=a[1]; liste[5][5]=e[5];
true
true
true
gap> liste[3][2]*liste[1][1];
130
gap> liste[4][5]/liste[3][5];
30/19
gap> liste[1][1]+liste[2][2];
21
```

```
gap> liste[2][5]-liste[3][1];
3
```

`Position()` komutu girilen değerin liste içerisinde yer alıp almadığını, alıyorsa listenin kaçıncı elemanı olduğunu gösterir. Eğer girilen değer liste içerisinde yer almıyorsa GAP programı `fail` yanıtı verecektir [17].

```
gap> Position(asal,17);
7
gap> Position(asal,71);
20
gap> Position(asal,42);
fail
```

GAP programında `Concatenation()` komutu iki listenin birleştirilip tekbir liste halinde yazılmasını sağlar. `Concatenation()` komutu kullanılarak birleştirilen iki liste aynı elemanları içeriyorsa bu elemanlar oluşacak yeni listede iki defa yazılacaktır [17].

```
gap> A:=[1,2,3,4];
[ 1, 2, 3, 4 ]
gap> B:=[5,6,7,8];
[ 5, 6, 7, 8 ]
gap> C:=Concatenation(A,B);
[ 1, 2, 3, 4, 5, 6, 7, 8 ]
gap> D:=[1,2,3,4,5,6,7,8,9];
[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
F:=Concatenation(C,D);
[ 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
```

GAP programında `Factors()` komutu bir sayıyı asal çarpanlarına ayırmak için kullanılır. Bu komut kullanıldığında `()` işaretleri arasına yazılan sayının asal çarpanları bir liste içersinde verilecektir. `()` işaretleri arasına pozitif olmayan bir sayı yazılırsa bulunan asal çarpanlardan ilki negatif sayı olarak yazılır [17].

```
gap> Factors(10);
[ 2, 5 ]
```

```
gap> Factors(12);
[ 2, 2, 3 ]
gap> Factors(-12);
[ -2, 2, 3 ]
gap> Factors(-1322);
[ -2, 661 ]
gap> Factors(1462);
[ 2, 17, 43 ]
```

GAP programında bir liste içerisinde aynı elemandan kaç tane olduğu `Collected()` komutu kullanılarak hesaplanabilir. Bu komut kullanıldığında verilen listenin tüm elemanları ikililer halinde (kendisi ve liste içerisinde kaç tane olduğu) yazılır [17].

```
gap> liste:=[1];
[ 1 ]
gap> Collected(liste);
[ [ 1, 1 ] ]
gap> Collected(A);
[ [ 1, 1 ], [ 2, 1 ], [ 3, 1 ], [ 4, 1 ] ]
gap> Collected(F);
[ [ 1, 2 ], [ 2, 2 ], [ 3, 2 ], [ 4, 2 ], [ 5, 2 ], [ 6, 2 ],
[ 7, 2 ], [ 8, 2 ], [ 9, 1 ] ]
```

`Factors()` komutu kullanılarak bulunan asal bölenlerin, kaçar tane olduğu da `Collected()` komutu kullanılarak bulunabilir [17].

```
gap> a:=Factors(Factorial(13));
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 5, 5, 7, 11, 13 ]
gap> Collected(a);
[ [ 2, 10 ],[ 3, 5 ],[ 5, 2 ],[ 7, 1 ],[ 11, 1 ],[ 13, 1 ] ]
gap> Collected(Collected(a));
[ [[ 2, 10 ], 1],[[ 3, 5 ], 1],[[ 5, 2 ], 1],[[ 7, 1 ],1],
[[ 11, 1 ], 1], [[ 13, 1 ], 1]]
```

GAP programında `Flat()` komutu kullanılarak ikililer halinde yazılmış bir liste düz liste halinde düzenlenir [17].

```
gap> Flat([1,[2,3],[[1,2],3]]);
[ 1, 2, 3, 1, 2, 3 ]
```

GAP programında `Reversed()` komutu bir listeyi tersine çevirmek için kullanılır. Bunun anlamı yeni listede listenin ilk elemanı son, son elemanı ise ilk eleman olacaktır [17].

```
gap> A:=[1,2,3,4,5,6,7,8,9];
[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
gap> Reversed(A);
[ 9, 8, 7, 6, 5, 4, 3, 2, 1 ]
gap> B:=[];
[ ]
gap> Reversed(B);
[ ]
gap> Reversed(B)=B;
true
```

GAP programında bir listenin en büyük elemanı `Maximum()` komutu kullanılarak en küçük elemanı ise `Minimum()` komutu kullanılarak bulunur. Liste elemanları teklilerden oluşuyorsa büyüklük veya küçüklük kıyaslaması yapmak kolaydır. Liste elemanları çiftlilerden (veya daha fazla) oluşuyor ise büyüklük ve küçüklük kıyaslaması yapmak için ilk önce ikililerin (veya daha fazla) ilk elemanları incelenir. Bu elemanlar içerisinde en küçük olan minimum değeri en büyük olan maksimum değeri verir. Birden fazla ikilinin (veya daha fazla) ilk elemanları aynı ise minimum ve maksimum değer bir sonraki elemanların büyüklük karşılaştırması yapılarak hesaplanır.

Bir liste içerisinde hem tekli hemde çiftli (veya daha fazla) elemanlar kullanılmışsa minimum va maksimum değeri hesaplamak için yine ilk elemanlara bakılır. Bir ikilinin ilk elemanı ile tekli elaman aynı büyüklükte ise ikili eleman daha büyük olacaktır. Bir ikilinin ilk elemanı ile tekli eleman aynı küçüklükte ise tekli eleman daha küçük olacaktır. Liste tekliler ve daha fazla n-lilerden oluşuyorsa büyüklük ve küçüklük karşılaştırması benzer şekilde yapılacaktır.

```
gap> Maximum(A);
9
```

```
gap> Minimum(A);
1
gap> K:=[[2,0],[3,1],[9,11],[7,-25],[9,12],[2,3]];
[[ 2, 0 ],[ 3, 1 ],[ 9, 11 ],[ 7, -25 ],[ 9, 12 ],[ 2, 3 ]]
gap> Maximum(K);
[ 9, 12 ]
gap> Minimum(K);
[ 2, 0 ]
gap> L:=[2,[2,0],[3,1],[9,11],[7,-25],[9,12],[2,3],9];
[2,[ 2, 0 ],[ 3, 1 ],[ 9, 11 ],[ 7, -25 ],[ 9, 12 ],[ 2, 3 ],9]
gap> Maximum(L);
[ 9, 12 ]
gap> Minimum(L);
2
```

GAP programında listelerin her bir elemanına bir özellik kazandırmak için `List()` komutu kullanılır. `()` işaretleri arasına sırası ile liste ve bu liste elemanlarına farklı bir özellik katmak için yazılacak fonksiyon kullanılmalıdır [17].

```
gap> List([1,2,3],i -> 2*i);
[ 2, 4, 6 ]
gap> List([1,2,3],k -> k^4);
[ 1, 16, 81 ]
gap> List([1,2,3],i -> Factorial(i));
[ 1, 2, 6 ]
```

Yukarıdaki örnekte `i` (veya herhangi bir değişken) değişkeni verilen listenin her bir elemanını temsil etmektedir. Liste elemanlarına bir özellik verirken tanımlı GAP programı fonksiyonları kullanılıyorsa (`Factorial,IsPrime` vb.) gerçekte `i` gibi bir değişken kullanılması zorunlu olmamakla beraber kullanıldığı durumda da GAP programı hata vermez.

```
gap> List([1,2,3],Factorial);
[ 1, 2, 6 ]
gap> List([1..9],IsPrime);
[ false, true, true, false, true, false, true, false, false ]
```

GAP programında bir listenin belirli bir özelliği taşıyan elemanlarını seçerek yeni bir liste oluşturmak için `Filtered()` komutu kullanılır. `()` işaretleri arasına sırası ile liste ve bu liste elemanlarından seçim yapılacak kural yazılmalıdır. Bu kural mutlaka `true` veya `false` biçiminde yanıt alınabilecek komutlardan oluşmalıdır. `true` yanıtı alacak elemanlar yeni listenin elemanlarını oluşturacaktır.

```
gap> Filtered([0..10],i -> i^2=i);
[ 0, 1 ]
gap> Filtered([1..30],IsPrime);
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ]
gap> A:=Group((1,2,3,4,5,6));
Group([ (1,2,3,4,5,6) ])
gap> Filtered(A,i-> Order(i)=3);
[ (1,5,3)(2,6,4), (1,3,5)(2,4,6) ]
```

Yukarıdaki örnekte `Filtered()` komutu kullanılarak grup elemanları içerisinde mertebeleri üçe eşit olanlar seçilerek bir liste oluşturulmuştur [17].

GAP programında bir listenin eleman sayısını bulmak için `Length()` komutu kullanılmıştı. GAP programında `Number()` komutuda bir listenin uzunluğunu bulmak için kullanılır. `Number()` komutu `Length()` komutundan farklı olarak `Filtered()` komutu ile oluşturulacak özel listelerin uzunluğunu da `Filtered()` komutu kullanmadan bulmak için kullanılabilir.

```
gap> Number([2,3,5,7,8,9]);
6
gap> Number([1..1000],IsPrime);
168
gap> Number(A,i-> Order(i)=3);
2
```

GAP programında bir liste içerisinde istenilen özelliği taşıyan ilk elemanı bulmak için `First()` komutu kullanılır [17].

```
gap> First([10^5..10^8],IsPrime);
100003
```

```
gap> First([10^5..10^8],i -> not IsPrime(i) and i>120000);
120001
gap> First([1..100],i -> i<0);
fail
gap> First(A,i-> Order(i)=3);
(1,5,3)(2,6,4)
```

GAP programında `ForAll()` komutu kullanarak bir listenin tüm elemanlarının belirli bir özelliği taşıyıp taşımadığı test edilebilir [17].

```
gap> ForAll([1..100],IsPrime);
false
gap> ForAll([2,3,5,7,11],IsPrime);
true
gap> ForAll(A,i-> Order(i)=3);
false
```

GAP programında `ForAny()` komutu kullanarak bir listenin herhangi bir elemanının belirli bir özelliği taşıyıp taşımadığı test edilebilir [17].

```
gap> ForAny([1..100],IsPrime);
true
gap> ForAny([4,6,8,10,12,14],IsPrime);
false
gap> ForAny(A,i-> Order(i)=3);
true
```

GAP programında `Sum()` komutu bir listenin tüm elemanlarını sırayla veya belli bir özelliğe göre toplamak için kullanılır [17].

```
gap> Sum([2,3,5,42,45,51,52,53,56,69,71]);
449
gap> Sum([1..100]);
5050
gap> Sum([1..100],i -> i^3);
25502500
```

GAP programında `Product()` komutu bir listenin tüm elemanlarını sırayla veya belli bir özelliğe göre çarpmak için kullanılır [17].

```
gap> Product([2,3,5,42,45,51,52,53,56,69,71]);
2186391421468800
gap> Product([1..21]);
51090942171709440000
gap> Product([1..10],i -> i^3);
477847258398720000000
```

GAP programında `Iterated()` komutu bir listenin tüm elemanlarını sırayla ikililer halinde alır ve verilen fonksiyon altında inceler. Örnek olarak `Lcm()` komutu ile iki sayının en küçük ortak katını hesapladığı söylenebilir [17].

```
gap> Iterated([4,5,6,12],Lcm);
60
gap> Lcm(4,5);
20
gap> Lcm(20,6);
60
gap> Lcm(60,12);
60
```

Yukarıda verilen örnekte liste elemanlarının ilk ikisinden başlayarak OKEK leri alınmaya başlanmıştır.

4.5 GAP Programında Fonksiyon Kullanımı

GAP programında özel fonksiyonlar da oluşturulabilir. Fonksiyonlar oluşturulurken üç farklı yöntem kullanılabilir. Birinci yöntemde fonksiyon oluşturabilmek için -> simgesi kullanılabilir. Örnek olarak, yazılan sayıların dördüncü kuvvetini hesaplayan bir fonksiyon ;

```
gap> kuvvet4:=x -> x^4;
function( x ) ... end
gap>
```

biçiminde yazılabilir. Yukarıdaki komutlar yardımıyla yazılan fonksiyonu aşağıdaki gibi kullanılır [16].

```
gap> kuvvet4(2);
16
gap> kuvvet4(4);
256
```

GAP programında fonksiyon oluşturmak için kullanılan ikinci yöntem ise `function` komutudur [17].

```
gap> fonk:=function(x)
> local y;
> y:=x^4+2*x;
> return y;
> end;
function( x ) ... end
```

Yukarıdaki kodlar $y=f(x)=x^4+2x$ şeklindeki bir fonksiyonu hesaplamak için yazılmış kodlardır. Buradaki `function(x)` komutu yazılacak fonksiyonun temel parametresinin x olacağını göstermektedir. Fonksiyon tanımlanırken kullanılacak diğer değişkenler yani yerel değişkenler de `local` komutundan sonra belirtilmelidir. `return return` komutu ise fonksiyon içerisinde tanımlanan işlemler gerçekleştirildikten sonra geriye bir değer gönderilmesini sağlar.

```
gap> fonk(10);
10020
gap> fonk(1);
3
gap> fonk(2);
20
```

GAP programı kullanırken bir kez yazılmış bulunan bir fonksiyon daha sonra tekrar kullanılabilmektedir. Bu işlemi gerçekleştirmek için yazılan fonksiyonu bir şekilde kaydetmek gerekir. `LogTo` komutunda GAP programı kullanırken yapılan işlemlerin bir kaydının tutulduğundan daha önce bahsedilmişti, ancak bu komut yazılan fonksiyonu tekrar çağırmak

için yeterli değildir. `InputLogTo` komutu oluşturulan fonksiyonların tekrar yüklenmesine olanak sağlar.

```
gap> InputLogTo("fonk");
gap> fonk:=function(x)
> local y;
> y:=x^4+2*x;
> return y;
> end;
function( x ) ... end
gap> InputLogTo();
```

Yukarıdaki `InputLogTo("fonk");` komutu kendisinden sonraki komutların "`fonk`" adlı bir dosya içerisinde saklanılması sağlar. Bu dosya GAP programının kurulduğu (Muhtemelen gap4r3) dizinin içerisine oluşturulacaktır. GAP programından çıkıp oluşturulan "`fonk`" adlı dosya bir metin editörü ile açılırsa bu dosyanın içeriği aşağıdaki gibi olacaktır.

```
fonk:=function(x)
local y;
y:=x^4+2*x;
return y;
end;
InputLogTo();
```

Bu dosyada `InputLogTo();` satırı silinir ve dosya tekrar kaydedilirse daha sonra bu yazılan fonksiyon ne zaman kullanılmak istenirse `Read("DosyaAdı");` komutu kullanılarak dosya GAP programına okutulabilir [17].

```
gap> Read("fonk");
gap> fonk(1);
3
gap> fonk(3);
87
gap> fonk(1234567);
232305052922195258381425 5
```

GAP programında fonksiyon oluşturmanın üçüncü ve en kullanışlı yöntemi ise direkt olarak metin editörleri kullanarak `gi` dosyaları oluşturmaktır. `gi` dosyaları içerisinde birden fazla fonksiyon barındırabilen ve doğrudan GAP programı tarafından okunabilen dosyalardır. `gi` dosyaları oluşturulduktan sonra GAP programı `Read("fonk.gi")` şeklindeki bir komutla karşılaştığında `fonk.gi` dosyasının içerisinde yer alan her fonksiyonun kullanılabilir duruma geldiğini anlar.

Aşağıda kodları verilen fonksiyonu bir metin editörü içine yazarak `kuvvet.gi` adıyla kaydedilebilir. Fonksiyon verilen bir x sayısı için x^4+2x değerini hesaplayacaktır.

```
kuv:=function(x)
local y;
y:=x^4+2*x;
return y;
end;
```

Şimdi var olan `kuvvet.gi` dosyasını `Read` komutu kullanarak GAP programına okutulup ve daha sonra bu dosya içerisindeki fonksiyon kullanılabilir.

```
gap> Read("kuvvet.gi");
gap> kuv(2);
20
gap> kuv(10);
10020
```

`Read()` komutu bir dosyanın GAP programına okutulup çalışmaya hazır hale getirilmesi için kullanılır. `()` işaretleri arasına dosyanın adı ve varsa uzantısı yazılmalıdır. `ReadAsFunction()` komutu da yine GAP programına bir dosya okutmak için kullanılabilir. `ReadAsFunction()` komutu, kullanılacak olan dosyada ki fonksiyon için bir fonksiyon adı verilmesine, `function` ve `end` komutlarının bulunmasına ihtiyaç yoktur. Bunun anlamı metin içersindeki ifadeler `local` satırı ile başlar.

Aşağıda verilen program kodları bir metin editörü içinde oluşturulabilir ve bu dosya `carp.gi` adıyla kaydedilebilir.

```
local y;
y:=12;
return y*100;
```

Oluşturulmuş olan carp.gi dosyası ReadAsFunction komutu kullanılarak GAP programına okutulabilir ve aynı anda fonksiyon olarak kullanılabilir.

```
gap> ReadAsFunction("carp.gi")();
1200
```

Aşağıda kodları verilen fonksiyonu bir metin editörü içinde oluşturulabilir ve bu dosya tekcift.gi adıyla kaydedilebilir. Yazılan fonksiyon bir sayının tek mi yoksa çift mi olduğunu inceleyecektir.

```
Tekcift:=function(x)
local k;
k:=(-1)^x;
if x=0 then
Print(x," sayisi tek veya cift ozellige sahip degildir. \n");
elif k=1 then
Print(x," sayisi cift sayidir. \n");
else
Print(x," sayisi tek sayidir. \n");
fi;
end;
```

Oluşturulmuş olan tekcift.gi dosyası Read komutu kullanılarak GAP programına okutulabilir ve daha sonra bu dosya içerisindeki Tekcift isimli fonksiyon kullanılabilir.

```
gap> Read("tekcift.gi");
gap> Tekcift(46);
46 sayisi cift sayidir.
gap> Tekcift(23);
23 sayisi tek sayidir.
gap> Tekcift(0);
0 sayisi tek veya cift ozellige sahip degildir.
```

GAP programı kullanarak yarıçapı verilen bir dairenin alanını hesaplayan bir fonksiyon oluşturulabilir. Metin editörü kullanarak `daire.gi` adı altında bir dosya oluşturulabilir. Yazdılan fonksiyon πr^2 değerini hesaplayıp ekrana yazdıracaktır.

```
daire:=function(r)
local pi,alan;
pi:=22/7;
alan:=pi*r^2;
return alan;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `daire.gi` dosya GAP programına okutulabilir ve içerisinde yer alan `daire` isimli fonksiyonu kullanarak herhangi bir r değeri için daire alanını hesaplanabilir.

```
gap> Read("daire.gi");
gap> daire(2);
88/7
gap> daire(112);
39424
```

GAP programı kullanarak yarıçapı verilen bir silindirin alanını hesaplayan fonksiyon yazılabilir. Metin editörü kullanarak `silindir.gi` adı altında bir dosya oluşturulabilir. Yazılan fonksiyon $2\pi r(r+h)$ değerini hesaplayıp ekrana yazdıracaktır.

```
AlanSilindir := function(arg)
local narg, alan, dikkat, hata, pi, r, h;
narg:= Length(arg);
dikkat:= " Dikkat: yarıcap ve yukseklik olmak uzere iki deger
girilmelidir. \n";
hata:= " Hata: Yarıcap ve yukseklik degerleri pozitif olmalıdır.
\n";
if ((narg < 2) or (narg >= 3)) then
Print(dikkat);
return false;
fi;
```

```
r:=arg[1];
h:=arg[2];
pi:=22/7;
alan:=(2*pi*r)*(r+h);
if (arg[1]<=0) or (arg[2]<=0) then
Print(hata);
return false;
fi;
return alan;
end;
```

Fonksiyon oluşturma esnasında kullanılan `narg` değişkeni, fonksiyon içerisinde listesi verilen elemanların uzunluğunu bulur. Oluşturulan fonksiyon için elemanların uzunluğu iki olmalıdır çünkü bir silindirin alanını bulmak için yarıçap ve yükseklik değerlerine ihtiyaç vardır. GAP programı kullanıcısı `AlanSilindir` fonksiyonunu kullanırken ikiden fazla veya eksik değer girerse yazılan fonksiyon uygun bir hata mesajı verecektir. Bir silindirin alanı hesaplanırken yarıçap ve yükseklik değerleri sıfır veya sıfırdan küçük bir sayı olamayacağından fonksiyon oluşturulurken girilen değerlerin pozitif sayı olup olmadığını denetleyen ve uygun mesajla ekrana yazdıran bir komutta hata değişkeni ile eklenebilir.

Yukarıdaki kodlar yazılarak oluşturulan `silindir.gi` adlı dosya GAP programına okutulabilir ve içerisinde yer alan `AlanSilindir` isimli fonksiyon kullanılarak herhangi bir r ve h değerleri için silindirin alanını hesaplanabilir.

```
gap> Read("silindir.gi");
gap> AlanSilindir(13,40);
30316/7
gap> AlanSilindir(7,21);
1232
gap> AlanSilindir(3,6,6);
 Dikkat: yarıcap ve yukseklik olmak uzere iki deger
girilmelidir.
false
gap> AlanSilindir(3,-2);
 Hata: Yarıcap ve yukseklik degerleri pozitif olmalıdır.
```

```
false
```

Bir sayının pozitif tam bölenlerinin toplamı, sayının iki katı oluyorsa bu sayıya mükemmel sayı denilmektedir. Örneğin 6 sayısı mükemmel bir sayıdır çünkü $1+2+3+6=12=2\times 6$ dır. GAP programı kullanarak verilen bir sayının mükemmel bir sayı olup olmadığını hesaplayan bir fonksiyon oluşturulabilir. Metin editörü kullanılarak `mükemmel.gi` adı altında bir dosya oluşturulabilir.

```
Mukemmel:=function(n)
local mu,i,top,kat;
top:=0;
kat:=2*n;
for i in [1..n] do
mu:=(n mod i);
if (mu=0) then
top:=top+i;
fi;
od;
if (kat=top) then
Print(" ",n ," sayisi mukemmel sayidir.\n");
else
Print(" ",n ," sayisi mukemmel sayi degildir.\n");
fi;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `mukemmel.gi` dosyası GAP programına okutulabilir ve içerisinde yer alan `Mukemmel` isimli fonksiyon kullanılarak herhangi bir *n* sayısının mükemmel sayı olup olmadığı incelenebilir.

```
gap> Read("mukemmel.gi");
gap> Mukemmel(6);
 6 sayisi mukemmel sayidir.
gap> Mukemmel(12);
 12 sayisi mukemmel sayi degildir.
gap> Mukemmel(10001224);
```

```
10001224 sayisi mukemmel sayi degildir.
```

Birden büyük olmak üzere kendisinden ve birden başka tam böleni olmayan sayılara asal sayılar denir. GAP programı kullanılarak verilen bir n sayısından küçük olan kaç tane asal sayı olduğunu hesaplayan bir fonksiyon oluşturulabilir. (Örnek olarak 10 sayısından küçük 4 tane asal sayı olup bunlar 2,3,5 ve 7 dir.) Metin editörü kullanarak kacasal.gi adı altında bir dosya oluşturulabilir.

```
Kacasal:=function(n)
local i,j,kactane,asalmi;
kactane:=0;
i:=n-1;
while i>1 do
asalmi:=1;
j:=2;
while j<i do
if (i mod j=0) then
asalmi:=0;
fi;
j:=j+1;
od;
if (asalmi=1) then
kactane:=kactane+1;
fi;
i:=i-1;
od;
return kactane;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan kacasal.gi dosyası GAP programına okutulabilir ve içerisinde yer alan Kacasal isimli fonksiyon kullanılarak herhangi bir n sayısından küçük kaç tane asal sayı olduğu hesaplanabilir.

```
gap> Read("kacasal.gi");
gap> Kacasal(2);
0
```

```
gap> Kacasal(3);
1
gap> Kacasal(10);
4
gap> Kacasal(100);
25
```

5 KÜMELER

5.1 Giriş

Bu bölümde küme ve küme ile ilgili özellikler, tanımlar ve teoremler GAP programı örnekleri ile sunulmuştur. Teoremlerin matematiksel ispatları yerine de GAP programı aracılığı ile yapılan ispatlara yer verilmiştir. Böylelikle matematiksel ifadelerin programın kullanılması amaçlanmıştır.

Bu bölümde kümelerde çok iyi bilinen küme elemanı, elemanın kümeye ait olup olmaması, eşit kümeler, boş küme, alt küme, kümelerin arakesiti, birleşimi, farkı, kartezyen çarpımları, simetrik farkları kavramları matematiksel olarak verilmiş ve GAP programı örnekleriyle incelenmiştir.

5.2 Küme Kavramı

Küme nesnelerin bir topluluğudur. Kümeyi oluşturan nesnelere kümenin elemanları denir [12].

GAP programında kümeler listeler yardımıyla oluşturulurlar. Bir listenin bir küme oluşturup oluşturmaması liste elemanlarının doğru dizilimine bağlıdır. Örnek olarak "iller" adında bir liste oluşturalım. Oluşturulan listenin bir küme olup olmadığı `IsSet()` fonksiyonu ile sorgulanır. Oluşturulan liste elemanları harf sırasına göre düzenlenmediğinden bir küme oluşturmayacaktır. Bir listenin elemanlarının bir küme oluşturması ancak `Set()` GAP fonksiyonunun kullanılması ile olacaktır [17].

```
gap> iller:=["ankara","corum","bursa"];
[ "ankara", "corum", "bursa" ]
gap> IsSet(iller);
false
gap> A:=Set(iller);
[ "ankara", "bursa", "corum" ]
gap> IsSet(A);
true
```

Bir liste içerisinde harf ve rakamların bir arada kullanılabileceği açıktır. Bu şekildeki listelerden de bir küme oluşturulabilir.

```
gap> kutahya:=["k","u","t","a","h","y","a",4,3];
[ "k", "u", "t", "a", "h", "y", "a", 4, 3 ]
gap> IsSet(kutahya);
false
gap> A:=Set(kutahya);
[ 3, 4, "a", "h", "k", "t", "u", "y" ]
```

Bir liste içerisinde bir eleman birden fazla kullanılmışsa, bu listenin küme haline getirilmesi durumunda bu elemanlardan yalnızca bir tanesi küme elemanları içerisinde yer alacaktır.

```
gap> K:=[1,2,3,4,1,3,"kutahya","eskisehir","corum","kutahya"];
[ 1, 2, 3, 4, 1, 3, "kutahya", "eskisehir", "corum", "kutahya" ]
gap> IsSet(K);
false
gap> KK:=Set(K);
[ 1, 2, 3, 4, "corum", "eskisehir", "kutahya" ]
```

Bir elemanın bir küme içerisinde yer alıp almadığı "in" komutu ile sorgulanır.

```
gap> "ankara" in A;
true
gap> "denizli" in A;
false
```

Bir kümeye eleman eklemek için AddSet() fonksiyonu kullanılır. Buradaki () işaretleri arasına eklenecek olan eleman ve küme adı yazılmalıdır [17].

```
gap> AddSet(A,"eskisehir");
gap> A;
["ankara", "bursa", "corum", "eskisehir"]
gap> AddSet(A,"denizli");
gap> A;
["ankara","bursa","corum","denizli","eskisehir"]
```

Burada dikkat edilecek önemli bir nokta ise eklenen elemanın küme içerisine uygun sırada yerleşmiş olmasıdır.

Bir kümeden eleman çıkarmak için `RemoveSet()` fonksiyonu kullanılır. Buradaki `()` işaretleri arasına çıkarılacak olan eleman ve küme adı yazılmalıdır [17].

```
gap> RemoveSet(A,"denizli");
gap> A;
[ "ankara", "bursa", "corum", "eskisehir" ]
```

Şimdi rakamlar yardımıyla kümeler oluşturalım ve işlemler yapalım.

```
gap> tek:=[1,5,7,3,9,11];
[ 1, 5, 7, 3, 9, 11 ]
gap> A:=Set(tek);
[ 1, 3, 5, 7, 9, 11 ]
gap> cift:=[2,6,8,4,10,12];
[ 2, 6, 8, 4, 10, 12 ]
gap> B:=Set(cift);
[ 2, 4, 6, 8, 10, 12 ]
gap> AddSet(A,13);
gap> A;
[ 1, 3, 5, 7, 9, 11, 13 ]
gap> AddSet(B,14);
gap> B;
[ 2, 4, 6, 8, 10, 12, 14 ]
gap> RemoveSet(A,13);
gap> A;
[ 1, 3, 5, 7, 9, 11 ]
gap> RemoveSet(B,14);
gap> B;
[ 2, 4, 6, 8, 10, 12 ]
```

Bir kümeye tekbir eleman yerine, bir liste halinde elemanlar eklemek veya çıkarmak için `UniteSet()` ve `SubtractSet()` fonksiyonları kullanılabilir. Buradaki

fonksiyonlarda () işaretleri arasına eklenecek olan elemanlar liste halinde ve eklenecek küme yazılmalıdır [17].

```
gap> UniteSet(A,[15,13]);
gap> A;
[ 1, 3, 5, 7, 9, 11, 13, 15 ]
gap> SubtractSet(A,[13,15]);
gap> A;
[ 1, 3, 5, 7, 9, 11 ]
```

`IntersectSet()` fonksiyonu bir küme ve o küme ile birlikte verilen bir listenin ortak elemanlarını alarak verilen kümeyi bu elemanlardan oluşur hale getirir [17].

```
gap> IntersectSet(A,[5,3,11,12,15]);
gap> A;
[ 3, 5, 11 ]
```

Burada bahsedilen üç fonksiyon (`UniteSet()` , `SubtarctSet()` ve `IntersectSet()`) aslında daha sonra bahsedilecek olan birleşim, fark ve kesişim özelliklerinin verilen küme üzerine kısıtlanmışlarıdırlar [17].

Tanım : A ve B kümeleri verilmiş olsun. A kümesinin bütün elemanları B içinde ve B kümesinin bütün elemanları A içindeyse, bu kümelere eşit kümeler denir [19].

Bu tanımlama GAP programı aracılığı ile basit bir şekilde yapılır. `IsEqualSet()` fonksiyonu () işaretleri arasında verilecek olan iki listenin eşitliğini kontrol eder. Burada verilen aynı elemanlı listelerin dizilişleri farklı olsa da bu fonksiyon "`true`" sonucunu verecektir. Bunun nedeni bu fonksiyon içerisinde verilen listeler kıyaslanmadan önce kümeye dönüştürülmesidir. `IsEqualSet(list1,list2)` şeklindeki bir fonksiyon aslında `Set(list1)=Set(list2)` sorgulamasını gerçekleştirir [17].

```
gap> list1:=[1,2,3,5,8,12,25];
[ 1, 2, 3, 5, 8, 12, 25 ]
gap> list2:=[2,3,5,25,1,12,8];
[ 2, 3, 5, 25, 1, 12, 8 ]
gap> IsEqualSet(list1,list2);
true
```

Tanım : Hiçbir elemanı olmayan kümeye boş küme denir ve $\varnothing$ ile gösterilir [7]. Boş küme GAP programında [ ] şeklinde gösterilir.

Tanım : A ve B iki küme olsunlar. B nin her elemanı, A nın da bir elemanı ise B ye, A nın bir alt kümesi denir ve $B \subset A$ ile gösterilir. Her kümenin $\varnothing$ ve kendisi olmak üzere en az iki tane alt kümesi vardır [7].

GAP programında `Combinations()` komutu bir kümenin tüm alt gruplarını belirlemek için kullanılır. Buradaki `()` işaretleri arasına alt kümeleri bulunacak olan küme yazılmalıdır [17].

```
gap> L:=["a","b","c","d","e"];
[ "a", "b", "c", "d", "e" ]
gap> K:=Set(L);
[ "a", "b", "c", "d", "e" ]
gap> AltK:=Combinations(K);
[[ ],["a"],["a","b"],["a","b","c"],["a","b","c","d"],
["a","b" "c","d","e"],["a","b","c","e"],["a","b","d"],
["a","b","d","e"],["a","b","e"],["a","c"],["a","c","d"],
["a","c","d","e"],["a","c","e"],["a","d"],["a","d","e"],
["a","e"],["b"],["b","c"],["b","c","d"],["b","c","d","e"],
["b","c","e"],["b","d"],["b","d","e"],["b","e"],["c"],["c","d"],
["c","d","e"],["c","e"],["d"],["d","e"],["e"]]
gap> M:=AltK[1];
[  ]
gap> IsSet(M);
true
gap> IsSubset(K,M);
true
gap> N:=AltK[5];
[ "a", "b", "c", "d" ]
gap> IsSet(N);
true
gap> IsSubset(K,N);
true
```

```
gap> O:=AltK[15];
[ "a", "d" ]
gap> IsSet(O);
true
gap> IsSubset(K,O);
true
gap> P:=AltK[28];
[ "c", "d", "e" ]
gap> IsSet(P);
true
gap> IsSubset(K,P);
true
```

GAP programında `NrCombinations()` komutu bir kümenin kaç tane alt kümesi olduğunu bulmak için kullanılır. Buradaki `()` işaretleri arasına alt küme sayısı bulunacak olan küme yazılmalıdır [17].

```
gap> NrCombinations(K);
32
gap> K:=[1];
[ 1 ]
gap> Combinations(K);
[ [  ], [ 1 ] ]
gap> NrCombinations(K);
2
```

`IsSubset()` fonksiyonu verilen bir kümenin diğerinin alt kümesi olup olmadığını test eder. Buradaki `()` işaretleri arasına ilk önce çok elemanı olan büyük küme sonra alt küme olup olmadığı test edilecek olan küme yazılmalıdır [17].

```
gap> A;
[ 1, 3, 5, 7, 9, 11 ]
gap> B;
[ 2, 4, 6, 8, 10, 12 ]
gap> C:=[1,5,11];
[ 1, 5, 11 ]
```

```
gap> IsSubset(A,C);
true
gap> IsSubset(A,B);
false
```

Teorem : A ve B iki küme olmak üzere

$$A = B \Leftrightarrow A \subset B \text{ ve } B \subset A$$

olmasıdır [7].

Bu teoremin doğruluğunu GAP programı yardımıyla inceleyelim

```
gap> C:=[12,17,21,52,71];
[ 12, 17, 21, 52, 71 ]
gap> D:=[12,17,21,52,71];
[ 12, 17, 21, 52, 71 ]
gap> C=D; IsSubset(C,D); IsSubset(D,C);
true
true
true
```

Tanım : A ve B kümelerinden her ikisinde de ortak olan elemanların oluşturduğu kümeye A ile B nin arakesiti denir ve $A \cap B$ ile gösterilir. Arakesit tanımı gereği $A \cap B \subset A$ ve $A \cap B \subset B$ olduğu açıktır [7].

Tanım : Ortak hiç bir elemanı bulunmayan iki kümeye ayrık kümeler denir [19].

GAP programında iki veya daha fazla kümenin arakesitleri `Intersection()` fonksiyonu yardımıyla hesaplanabilir. Bu fonksiyondaki `()` işaretleri arasına kesişimleri alınacak kümeler isimleriyle veya liste halinde verilmelidir. Aslında bu fonksiyonu kullanmak için yazılacak listelerin küme oluşturma zorunluluğu da yoktur yani iki listenin kesişimi de alınabilir [17].

```
gap> A;
[ 1, 3, 5, 7, 9, 11 ]
gap> B;
[ 2, 4, 6, 8, 10, 12 ]
gap> Intersection(A,B);
```

```
[ ]
gap> Intersection([12,5,9,13],[3,12,7,5,11]);
[ 5, 12 ]
```

Tanım : A ve B kümelerinden en az birine ait olan elemanların oluşturduğu kümeye A ile B in birleşimi denir ve $A \cup B$ ile gösterilir. Birleşim tanımı gereği $A \subset A \cup B$ ve $B \subset A \cup B$ olduğu açıktır [7].

GAP programında iki veya daha fazla kümenin birleşimleri `Union()` fonksiyonu yardımıyla hesaplanabilir. Bu fonksiyondaki `()` işaretleri arasına birleşimleri alınacak kümeler isimleriyle veya liste halinde verilmelidir. Yine bu fonksiyon kullanılarak herhangi iki listenin birleşimi de alınabilir [17].

```
gap> A:=[1,3,5,7,9,11];
[ 1, 3, 5, 7, 9, 11 ]
gap> B:=[2,4,6,8,10,12];
[ 2, 4, 6, 8, 10, 12 ]
gap> Union(A,B);
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ]
gap> Union([1,3,5],[5,7,8],[11,3,5],[]);
[ 1, 3, 5, 7, 8, 11 ]
```

Teorem : $A \subset B \Rightarrow A \cap B = A$ ve $A \subset B \Rightarrow A \cup B = B$ dir [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> A:=[1,2,3,5,6];
[ 1, 2, 3, 5, 6 ]
gap> B:=[1,2,3,4,5,6,8,12];
[ 1, 2, 3, 4, 5, 6, 8, 12 ]
gap> IsSubset(B,A); Intersection(A,B)=A;
true
true
gap> IsSubset(B,A); Union(A,B)=B;
true
true
```

Teorem : A, B, C kümeler olmak üzere,

i) $A \cup A = A,\ A \cap A = A$

ii) $A \cup B = B \cup A\ ,\ A \cap B = B \cap A$

iii) $A \cup (B \cup C) = (A \cup B) \cup C\ ,\ A \cap (B \cap C) = (A \cap B) \cap C$

özellikleri sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> A:=[5,7,12,35,74,112];
[ 5, 7, 12, 35, 74, 112 ]
gap> B:=[5,6,8,10,12,14,15];
[ 5, 6, 8, 10, 12, 14, 15 ]
gap> C:=[12,35,36,37,41];
[ 12, 35, 36, 37, 41 ]
gap> Union(A,A)=A; Intersection(A,A)=A;
true
true
gap> Union(A,B)=Union(B,A); Intersection(A,B)=Intersection(B,A);
true
true
gap> Union(A,Union(B,C))=Union(Union(A,B),C);
true
gap>
Intersection(A,Intersection(B,C))=Intersection(Intersection(A,B)
,C);
true
```

Teorem : A, B, C kümeler olmak üzere,

i) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (birleşimin arakesit üzerine dağılımı)

ii) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (arakesitin birleşim üzerine dağılma)

özellikleri sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap>Union(A,Intersection(B,C))=Intersection(Union(A,B),Union(A,C
));
true
gap>Intersection(A,Union(B,C))=Union(Intersection(A,B),Intersect
ion(A,C));
true
```

Tanım : A ve B iki küme olsun. B ye ait, fakat A ya ait olmayan elemanların oluşturduğu kümeye B nin A ile fark kümesi denir ve $B-A$ ile gösterilir [7].

GAP programında `Difference()` fonksiyonu bir kümenin diğer kümeden farkını hesaplar. Bu fonksiyondaki `()` işaretleri arasına farkı alınacak kümeler isimleriyle veya liste halinde verilmelidir. Bu fonksiyon $B-A$ kümesini oluşturacaktır.

```
gap> A;
[ 5, 7, 12, 35, 74, 112 ]
gap> B;
[ 5, 6, 8, 10, 12, 14, 15 ]
gap> B\A:=Difference(A,B);
[ 7, 35, 74, 112 ]
```

Teorem : A ve B herhangi iki küme olmak üzere,

$$(A-B)\cap(B-A)=\varnothing$$

olur [15].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> Difference(A,B);
[ 7, 35, 74, 112 ]
gap> Difference(B,A);
[ 6, 8, 10, 14, 15 ]
gap> BOS:=[];
[ ]
gap> Intersection(Difference(A,B),Difference(B,A))=BOS;
true
```

Tanım : A ve B herhangi iki küme olsun. $a \in A$ ve $b \in B$ olmak üzere (a,b) sıralı ikililerinin oluşturduğu kümeye.

$$A \times B = \{(a,b) : a \in A, b \in B\}$$

A ile B in kartezyen çarpımı denir ve $A \times B$ ile gösterilir. İkiden fazla kümenin kartezyen çarpımı da benzer şekildedir [7].

Tanım : İkililerin eşitliği ;

$$(a,b) = (c,d) \Leftrightarrow a = c \text{ ve } b = d$$

ile tanımlanır [7].

GAP programında `Cartesian()` fonksiyonu kümelerin Kartezyen Çarpımı kümesini oluşturmak için kullanılır. Buradaki `()` Kartezyen Çarpım kümeleri oluşturulacak olan kümeler yazılmalıdır. Yine bu kümelerin ismi yazılabileceği gibi kümeler liste halinde de yazılabilir [17].

Örnek : $A = \{a,b,c\}$, $B = \{1,2\}$ ise

$$A \times B = \{(a,1),(a,2),(b,1),(b,2),(c,1),(c,2)\} \text{ ve}$$

$$B \times A = \{(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)\}$$

özellikleri sağlanır [7].

Şimdi bu örneği GAP programı yardımıyla inceleyelim;

```
gap> Cartesian(A,B);
[ [ "a", 1 ], [ "a", 2 ], [ "b", 1 ], [ "b", 2 ], [ "c", 1 ], [
"c", 2 ] ]
gap> Cartesian(B,A);
[ [ 1, "a" ], [ 1, "b" ], [ 1, "c" ], [ 2, "a" ], [ 2, "b" ], [
2, "c" ] ]
```

Teorem : A,B,C birer küme olmak üzere

$$A \times (B \cup C) = (A \times B) \cup (A \times C)$$

eşitliği sağlanır [7].

Şimdi bu teoremi GAP programı kullanarak inceleyelim.

```
gap> A:=["a","b","c","d","e"];
```

```
[ "a", "b", "c", "d", "e" ]
gap> B:=[1,3,4,7,9];
[ 1, 3, 4, 7, 9 ]
gap> C:=[İ,5,17,21,33,41,15];
[ 1, 5, 17, 21, 33, 41, 15 ]
gap>Cartesian(A,Union(B,C))=Union(Cartesian(A,B),Cartesian(A,C))
;
true
```

Örnek : $A=\{a,b,c\}$, $B=\{1,2,3\}$, $C=\{x\}$ ve $D=\varnothing$ olmak üzere aşağıda verilen kümelerin tüm elemanlarını belirleyelim [15].

a) $A\times B$ b) $B\times A$

c) $A\times B\times C$ d) $A\times D$

```
gap> A:=["a","b","c"];
[ "a", "b", "c" ]
gap> B:=[1,2,3];
[ 1, 2, 3 ]
gap> C:=["x"];
[ "x" ]
gap> D:=[];
[  ]
gap> Cartesian(A,B);
[ [ "a", 1 ],[ "a", 2 ],[ "a", 3 ],[ "b", 1 ],[ "b", 2 ],
[ "b", 3 ],[ "c", 1 ],[ "c", 2 ],[ "c", 3 ] ]
gap> a:=Cartesian(A,B);
[ [ "a", 1 ],[ "a", 2 ],[ "a", 3 ],[ "b", 1 ],[ "b", 2 ],
[ "b", 3 ],[ "c", 1 ],[ "c", 2 ],[ "c", 3 ] ]
gap> b:=Cartesian(B,A);
[ [ 1, "a" ],[ 1, "b" ],[ 1, "c" ],[ 2, "a" ],[ 2, "b" ],
[ 2, "c" ],[ 3, "a" ],[ 3, "b" ],[ 3, "c" ] ]
gap> c:=Cartesian(A,B,C);
[ [ "a", 1, "x" ],[ "a", 2, "x" ],[ "a", 3, "x" ],
[ "b", 1, "x" ],[ "b", 2, "x" ],[ "b", 3, "x" ],[ "c", 1, "x" ]
,[ "c", 2, "x" ],[ "c", 3, "x" ] ]
```

```
gap> d:=Cartesian(A,D);
[  ]
```

Tanım : A ve B bir E kümesinin alt kümeleri olsunlar. $E-A=A'$ kümesine A nın (E nin içindeki) tümleyeni denir [7].

Teorem (De Morgan) : A ve B birer küme olsun.

i) $(A\cap B)'=A'\cup B'$

ii) $(A\cup B)'=A'\cap B'$

özellikleri sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> E:=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16];
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 ]
gap> A:=[1,3,5,7,9,11,12,13];
[ 1, 3, 5, 7, 9, 11, 12, 13 ]
gap> B:=[2,4,6,8,10,13,14,15];
[ 2, 4, 6, 8, 10, 13, 14, 15 ]
gap> TA:=Difference(E,A);
[ 2, 4, 6, 8, 10, 14, 15, 16 ]
gap> TB:=Difference(E,B);
[ 1, 3, 5, 7, 9, 11, 12, 16 ]
gap> Difference(E,Intersection(A,B))=Union(TA,TB);
true
gap> Difference(E,Union(A,B))=Intersection(TA,TB);
true
```

Örnek : A,B,C kümeler olmak üzere

$$A\times(B-C)=(A\times B)-(A\times C)$$

olduğunu gösterelim [7].

```
gap> A:=[12,13,14,16,18,19,22,35];
[ 12, 13, 14, 16, 18, 19, 22, 35 ]
gap> B:=[10,11,65,74,75,78];
[ 10, 11, 65, 74, 75, 78 ]
```

```
gap> C:=[1,3,5,7,11,19];
[ 1, 3, 5, 7, 11, 19 ]
gap> Cartesian(A,Difference(B,C));
[[12,10],[12,65],[12,74],[12,75],[12,78],[13,10],[13,65],[13,74]
,[13,75],[13,78],[14,10],[14,65],[14,74],[14,75],[14,78],[16,10]
,[16,65],[16,74],[16,75],[16,78],[18,10],[18,65],[18,74],[18,75]
,[18,78],[19,10],[19,65],[19,74],[19,75],[19,78],[22,10],[22,65]
,[22,74],[22,75],[22,78],[35,10],[35,65],[35,74],[35,75],[35,78]
]
gap> Cartesian(A,B);
[[12,10],[12,11],[12,65],[12,74],[12,75],[12,78],[13,10],[13,11]
,[13,65],[13,74],[13,75],[13,78],[14,10],[14,11],[14,65],[14,74]
,[14,75],[14,78],[16,10],[16,11],[16,65],[16,74],[16,75],[16,78]
,[18,10],[18,11],[18,65],[18,74],[18,75],[18,78],[19,10],[19,11]
,[19,65],[19,74],[19,75],[19,78],[22,10],[22,11],[22,65],[22,74]
,[22,75],[22,78],[35,10],[35,11],[35,65],[35,74],[35,75],[35,78]
]
gap> Cartesian(A,C);
[[12,1],[12,3],[12,5],[12,7],[12,11],[12,19],[13,1],[13,3],[13,5
],[13,7],[13,11],[13,19],[14,1],[14,3],[14,5],[14,7],[14,11]
,[14,19],[16,1],[16,3],[16,5],[16,7],[16,11],[16,19],[18,1]
,[18,3],[18,5],[18,7],[18,11],[18,19],[19,1],[19,3],[19,5],
[19,7],[19,11],[19,19],[22,1],[22,3],[22,5],[22,7],[22,11],
[22,19],[35,1],[35,3],[35,5],[35,7],[35,11],[35,19]]
gap>Cartesian(A,Difference(B,C))=Difference(Cartesian(A,B),Carte
sian(A,C));
true
```

Tanım : A ve B iki küme olmak üzere

$$A\Delta B = (A - B) \cup (B - A)$$

kümesine A ile B in simetrik farkı denir [7].

Örnek : A ve B iki küme olmak üzere,

$$A\Delta B = (A \cup B) - (B \cap A)$$

olduğunu gösterelim [7].

```
gap> A;
[ 12, 13, 14, 16, 18, 19, 22, 35 ]
gap> B;
[ 10, 11, 65, 74, 75, 78 ]
gap> ASB:=Union(Difference(A,B),Difference(B,A));
[ 10, 11, 12, 13, 14, 16, 18, 19, 22, 35, 65, 74, 75, 78 ]
gap> ASB=Difference(Union(A,B),Intersection(A,B));
true
```

Örnek : A bir küme olmak üzere

$$A \cup \varnothing = A \text{ ve } A \cap \varnothing = \varnothing$$

olduğunu gösterelim [15].

```
gap> A:=[10,11,12,13,14,15,16,17,18,19,20];
[ 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 ]
gap> BOS:=[];
[ ]
gap> Union(A,BOS)=A;
true
gap> Intersection(A,BOS)=BOS;
true
```

Örnek : A ve B bir küme olmak üzere,

$$A \cup B = B \cup A \text{ ve } A \cap B = B \cap A$$

olduğunu gösterelim [15].

```
gap> B:=[14,16,18,20,24,26,28,30];
[ 14, 16, 18, 20, 24, 26, 28, 30 ]
gap> Union(A,B)=Union(B,A);
true
gap> Intersection(A,B)=Intersection(B,A);
true
```

Örnek : A, B birer küme olmak üzere,

$$A \cup B = (A \cap B) \cup (A - B) \cup (B - A)$$

eşitliğinin sağlandığını gösterelim [15].

```
gap> Union(A,B);
[ 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 24, 26, 28, 30 ]
gap> Intersection(A,B);
[ 14, 16, 18, 20 ]
gap> Difference(A,B);
[ 10, 11, 12, 13, 15, 17, 19 ]
gap> Difference(B,A);
[ 24, 26, 28, 30 ]
gap> Union(A,B)=Union(Intersection(A,B),Difference(A,B),
Difference(B,A));
true
```

Örnek : A, B birer küme olmak üzere,

$$(A \cap B) - B = \varnothing$$

eşitliğinin sağlandığını gösterelim [15].

```
gap> Difference(Intersection(A,B),B);
[ ]
gap> Difference(Intersection(A,B),B)=BOS;
true
```

Örnek : A, B birer küme olmak üzere,

$$(A \cup B) - B = A - B$$

eşitliğinin sağlandığını gösterelim [15].

```
gap> Difference(Union(A,B),B);
[ 10, 11, 12, 13, 15, 17, 19 ]
gap> Difference(A,B);
[ 10, 11, 12, 13, 15, 17, 19 ]
gap> Difference(Union(A,B),B)=Difference(A,B);
true
```

Örnek : A, B, C birer küme olmak üzere

$$A-(B\cup C)=(A-B)\cap(A-C)$$

olduğunu gösterelim [15].

```
gap> X:=["a","b","c","e","f","g",5,10,15];
[ "a", "b", "c", "e", "f", "g", 5, 10, 15 ]
gap> Y:=["k","l","m","e","f","g",55,10,15];
[ "k", "l", "m", "e", "f", "g", 55, 10, 15 ]
gap> Z:=["c","d","e","f","k","l",10,20,30];
[ "c", "d", "e", "f", "k", "l", 10, 20, 30 ]
gap> A:=Set(X);
[ 5, 10, 15, "a", "b", "c", "e", "f", "g" ]
gap> B:=Set(Y);
[ 10, 15, 55, "e", "f", "g", "k", "l", "m" ]
gap> C:=Set(Z);
[ 10, 20, 30, "c", "d", "e", "f", "k", "l" ]
gap> Difference(A,Union(B,C));
[ 5, "a", "b" ]
gap> Intersection(Difference(A,B),Difference(A,C));
[ 5, "a", "b" ]
gap> Difference(A,Union(B,C))=Intersection(Difference(A,B),
Difference(A,C));
true
```

Örnek : A,B,C birer küme olmak üzere,

$$A\cap(B-C)=(A\cap B)-(A\cap C)$$

olduğunu gösterelim [15].

```
gap> Intersection(A,Difference(B,C));
[ 15, "g" ]
gap> Difference(Intersection(A,B),Intersection(A,C));
[ 15, "g" ]
gap> Intersection(A,Difference(B,C))=Difference(Intersection
(A,B),Intersection(A,C));
true
```

6 FONKSİYONLAR

Tanım : A ve B iki küme olsun. A nın her bir elemanını B nin bir elemanına eşleyen bir kurala A dan B ye bir fonksiyon denir ve $f : A \to B$ şeklinde gösterilir [2].

GAP programında `MappingByFunction()` komutu bir fonksiyon tanımlamak için kullanılır. `()` işaretleri arasına sırasıyla fonksiyonu oluşturulacak olan tanım kümesi , görüntü kümeleri bir grup olarak ve fonksiyonun dönüşümü girilmelidir [17].

```
gap> G:=Group((1,2,3),(1,2));
Group([ (1,2,3), (1,2) ])
gap> H:=Group((1,3,2),(1,3));
Group([ (1,3,2), (1,3) ])
gap> fonk1:=MappingByFunction(G,H,x->x^2);
MappingByFunction( Group([ (1,2,3), (1,2) ]), Group([ (1,3,2),
(1,3) ]), function( x ) ... end )
```

Burada bir konuya açıklık getirmek gerekmektedir. GAP programında `function` komutu kullanılarak fonksiyonlar tanımlanabilmekte idi. Burada bahsedilen fonksiyonların matematiksel fonksiyonlarla bir alâkası olmayıp sadece program parçalarıdır. Bir matematiksel yapının fonksiyon olup olmadığını test etmek için GAP programında `IsMapping()` komutu kullanılır [17].

```
gap> IsMapping(fonk1);
true # fonk1 yapısı bir fonksiyondur.
gap> IsMapping(x->2);
false # x->2 ifadesi GAP programında bir fonksyon anacak
matematiksel fonksiyon değildir.
```

Tanım : $f : A \to B$ fonksiyonun da A kümesine f fonksiyonun tanım kümesi , B kümesine f fonksiyonunun değer kümesi denir [19].

GAP programında bir fonksiyonun tanım kümesini bulmak için `Source()` komutu kullanılır [17].

```
gap> S:=Source(fonk1);
Group([ (1,2,3), (1,2) ])
gap> G=S;
true
```

GAP programında bir fonksiyonun değer kümesini bulmak için `Range()` komutu kullanılır [17].

```
gap> R:=Range(fonk1);
Group([ (1,3,2), (1,3) ])
gap> H=R;
true
```

Tanım : Bir $a \in A$ için f kuralı bu a elemanını B kümesinin $b \in B$ elemanına bağlıyorsa b ye a nın f altındaki görüntüsü denir ve $f(a)=b$ şeklinde gösterilir [19].

GAP programında tanım kümesinden alınan bir elemanın görüntüsünü bulmak için ^ üst alma operatörü kullanılabilir. Bu elemanın fonksiyon altındaki görüntüsü değer kümesinde bir eleman olmalıdır [17].

```
gap> eA:=Elements(A);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> eA[2]^fonk3;
(1,2,3,4)
gap> (1,2,3,4) in B;
true
gap> eA[3];
(1,3)(2,4)
gap> eA[3]^fonk3;
(1,3)(2,4)
gap> (1,3)(2,4) in B;
true
```

GAP programında tanım kümesinden alınan bir elemanın görüntüsünü bulmak için ^ üst alma operatöründen başka `Image()` komutu da kullanılabilir. `()` işaretleri arasına sırasıyla fonksiyon ve görüntüsü bulunacak olan eleman yazılmalıdır [17].

```
gap> Image(fonk3,eA[2]);
(1,2,3,4)
gap> Image(fonk3,eA[3]);
(1,3)(2,4)
```

Tanım : f ve g tanım ve değer kümeleri aynı kümeler olsun. Eğer tanım kümesindeki her x için $f(x)=g(x)$ ise f ve g fonksiyonlarına eşit fonksiyonlar denir [19].

GAP programında = karşılaştırma operatörü iki değerin eşitliğini sorgulamakla kalmaz iki fonksiyonun eşitliğini de inceler [17].

```
gap> A:=Group((1,2,3),(1,2));
Group([ (1,2,3), (1,2) ])
gap> B:=Group((1,3,2),(1,3));
Group([ (1,3,2), (1,3) ])
gap> fonk2:=MappingByFunction(A,B,x->x^2);
MappingByFunction( Group([ (1,2,3), (1,2) ]), Group([ (1,3,2),
(1,3) ]), function( x ) ... end )
gap> fonk1=fonk2;
true
```

Tanım : $f:X\to Y$ bir fonksiyon olmak üzere X in birbirinden farklı herhangi iki elemanın görüntüleri de farklıysa f fonksiyonuna birebir fonksiyon denir.

Başka bir ifadeyle X in her x,x' elemanları için $f(x)=f(x')$ eşitliği $x=x'$ eşitliğini gerektiriyorsa f fonksiyonuna birebir fonksiyon denir [19].

GAP programında bir fonksiyonun birebir olup olmadığı `IsInjective()` komutu kullanılarak bulunur [17].

```
gap> IsInjective(fonk1);
false
gap> A:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> B:=Group((2,3,4,1));
Group([ (1,2,3,4) ])
```

```
gap> fonk3:=MappingByFunction(A,B,x->x);
MappingByFunction( Group([ (1,2,3,4) ]), Group(
[ (1,2,3,4) ]), function( x ) ... end )
gap> IsInjective(fonk3);
true
```

Tanım : $f: X \to Y$ bir fonksiyon olmak üzere Y nin her y elemanı için X in $f(x) = y$ özelliğini sağlayan en az bir elemanı varsa (yani Y nin her elemanı X in en az bir elemanının görüntüsü ise) f fonksiyonuna örten fonksiyon denir [19].

GAP programında bir fonksiyonun örten olup olmadığı `IsSurjective()` komutu kullanılarak bulunur [17].

```
gap> IsSurjective(fonk1);
false
gap> IsSurjective(fonk3);
true
```

GAP programında bir fonksiyonun birebir ve örten olup olmadığı `IsBijective()` komutu kullanılarak bulunur [17].

```
gap> IsBijective(fonk1);
false
gap> IsBijective(fonk2);
true
```

Tanım : X bir küme olmak üzere $\forall x \in X$ için $Id_X(x) = x$ şeklinde tanımlanan $Id_X: X \to X$ fonksiyonuna X üzerinde özdeş fonksiyon denir [2].

GAP programında bir X kümesi üzerinde özdeş fonksiyon oluşturmak için `IdentityMapping()` komutu kullanılır. `()` işaretleri arasına özdeş fonksiyonu bulunacak olan grup yazılmalıdır [17].

```
gap> D:=Group((1,2,3,4),(1,2,3));
Group([ (1,2,3,4), (1,2,3) ])
gap> IdF:=IdentityMapping(D);
```

```
IdentityMapping( Group([ (1,2,3,4), (1,2,3) ]) )
gap> eD:=Elements(D);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> eD[5]^IdF=eD[5];
true
gap> eD[15]^IdF=eD[15];
true
```

Tanım : X ve Y iki küme ve $c \in Y$ olmak üzere $\forall x \in X$ için $f(x) = c$ şeklinde tanımlanan f fonksiyonuna X den Y ye c değerli sabit fonksiyon denir [19].

Sabit fonksiyon için kullanılan bir GAP programı fonksiyonu yoktur. Tanım kümesinin her elemanını değer kümesinin aynı elemanına götüren bir fonksiyon bilinen metotla yazılabilir. Bu fonksiyon sabit fonksiyon olacaktır.

```
gap> M:=Group((1,2,3,4),(1,2));
Group([ (1,2,3,4), (1,2) ])
gap> N:=Group((1,4,3,2),(1,2));
Group([ (1,4,3,2), (1,2) ])
gap> eN:=Elements(N);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4), (1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3) ]
gap> fonk4:=MappingByFunction(M,N,x->eN[3]);
MappingByFunction( Group([ (1,2,3,4), (1,2) ]), Group([
(1,4,3,2), (1,2) ]), function( x ) ... end )
```

Tanım : Bir $f : X \to Y$ fonksiyonu verilmiş olmak üzere bir $A \subseteq X$ alt kümesi için Y nin $f(A) = \{f(x) \mid x \in A\}$ olarak tanımlanan alt kümesine A nın f altındaki görüntüsü denir [19].

GAP programında $A \subseteq X$ için A nın f altındaki görüntüsü `Image()` komutu kullanılarak bulunur. `()` işaretleri arasına sırasıyla ilk önce görüntüsü bulunacak olan fonksiyon ve alt küme yazılmalıdır [17].

X kümesini kendisinin alt kümesi olduğundan bu küme içinde görüntü bulunabilir. `ImagesSource()` komutu $X \subseteq X$ alt kümesi için X in f altındaki görüntüsünü bulmak için kullanılabilir. Bu komut `Image()` komutu ile aynı sonucu verecektir. `Image()` komutu $X \subseteq X$ alt kümesinin görüntüsünü bulmak için kullanırsa yalnızca fonksiyon ismi vermek yeterli olacaktır. Eğer fonksiyon örtense $X \subseteq X$ alt kümesi için X in f altındaki görüntüsü fonksiyonun değer kümesini verecektir [17].

```
gap> eA;
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> AA:=Subgroup(A,[eA[3]]);
Group([ (1,3)(2,4) ])
gap> Image(fonk3,AA);
[ (), (1,3)(2,4) ]
gap> Image(fonk3);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap>g1:= Image(fonk3,A);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap>g2:= ImagesSource(fonk3);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap>g1=g2;
true
gap> g2=B;
true
```

Tanım : Bir $f : X \to Y$ fonksiyonu verilmiş olmak üzere bir $B \subseteq Y$ alt kümesi için X in $f^{-1}(B) = \{x \in X \mid f(x) \in B\}$ biçiminde tanımlanan alt kümesine B nın f altındaki ters görüntüsü denir [19].

GAP programında bir fonksiyonun tersi `InverseGeneralMapping()` komutu kullanılır. `()` işaretleri arasına tersi bulunacak olan fonksiyon yazılmalıdır [17].

```
gap> tfonk3:=InverseGeneralMapping(fonk3);
InverseGeneralMapping( MappingByFunction( Group([ (1,2,3,4) ]),
Group([ (1,2,3,4) ]), function( x ) ... end ) )
gap> eB:=Elements(B);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> BB:=Subgroup(B,[eB[4]]);
Group([ (1,4,3,2) ])
gap> Image(tfonk3,BB);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
```

Teorem : $f: M \to N$ fonksiyonu ve $M_1, M_2 \subseteq M$ ve $N_1, N_2 \subseteq N$ altkümeleri verilmiş olsun. Bu durumda

i) $f(M_1 \cup M_2) = f(M_1) \cup f(M_2)$

ii) $f(M_1 \cap M_2) \subseteq f(M_1) \cap f(M_2)$

iii) $f^{-1}(N_1 \cup N_2) = f^{-1}(N_1) \cup f^{-1}(N_2)$

iv) $f^{-1}(N_1 \cap N_2) = f^{-1}(N_1) \cap f^{-1}(N_2)$

özellikleri sağlanır [19].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> M:=Group((1,2,3));
Group([ (1,2,3) ])
gap> N:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> eM:=Elements(M);
[ (), (1,2,3), (1,3,2) ]
gap> eN:=Elements(N);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> MM:=Subgroup(M,[eM[2]]);
Group([ (1,2,3) ])
gap> MMM:=Subgroup(M,[eM[3]]);
Group([ (1,3,2) ])
gap> NN:=Subgroup(N,[eN[3]]);
Group([ (1,3)(2,4) ])
```

```
gap> NNN:=Subgroup(N,[eN[4]]);
Group([ (1,4,3,2) ])
gap> f4:=MappingByFunction(M,N,x->x^5);
MappingByFunction( Group([ (1,2,3) ]), Group(
[ (1,2,3,4) ]), function( x ) ... end )
gap> Image(f4,Union(MM,MMM))=Union(Image(f4,MM),Image(f4,MMM));
true
gap> A1:=Image(f4,Intersection(MM,MMM));
[ (), (1,2,3), (1,3,2) ]
gap> A2:=Intersection(Image(f4,MM),Image(f4,MMM));
[ (), (1,2,3), (1,3,2) ]
gap> IsSubset(A2,A1);
true
gap> Image(tf4,Union(NN,NNN))=Union(Image(tf4,NN),
Image(tf4,NNN));
true
gap> Image(tf4,Intersection(NN,NNN))=Intersection(Image(tf4,NN),
Image(tf4,NNN));
true
```

Tanım : $f: X \to Y$ ve $g: Y \to Z$ fonksiyonları verilsin. $h: X \to Z, h(x) = g(f(x))$ olarak tanımlanan h fonksiyonuna f ile g nin bileşke fonksiyonu denir ve $h = g \circ f$ şeklinde gösterilir [19].

GAP programında `CompositionMapping()` komutu iki fonksiyonun bileşkesini almak için kullanılır. `()` işaretleri arasına bileşkeleri alınacak olan fonksiyonlar yazılmalıdır. `IsCompositionMappingRep()` komutu bir fonksiyonun bileşke fonksiyon olup olmadığını test etmek için kullanılır [17].

```
gap> G:=Group((1,2,3));
Group([ (1,2,3) ])
gap> H:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> K:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
```

```
gap> f:=MappingByFunction(G,H,x->x);
MappingByFunction( Group([ (1,2,3) ]), Group(
[ (1,2,3,4) ]), function( x ) ... end )
gap> g:=MappingByFunction(H,K,x->x);
MappingByFunction( Group([ (1,2,3,4) ]), Group(
[ (1,2,3,4,5) ]), function( x ) ... end )
gap> h:=CompositionMapping(f,g);
CompositionMapping( MappingByFunction( Group([ (1,2,3) ]),
Group([(1,2,3,4) ]),function( x ) ... end ),
MappingByFunction( Group([ (1,2,3,4) ]), Group([ (1,2,3,4,5) ]),
function( x ) ... end ) )
gap> IsCompositionMappingRep(f);
false
gap> IsCompositionMappingRep(g);
false
gap> IsCompositionMappingRep(h);
true
```

Teorem : $f:X \to Y$, $g:Y \to Z$ ve $h:Z \to T$ fonksiyonları verilsin,

$$(h \circ g) \circ f = h \circ (g \circ f)$$

olur [19].

```
gap> X:=Group((1,2,3));
Group([ (1,2,3) ])
gap> Y:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> Z:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
gap> T:=Group((1,2,3,4,5,6));
Group([ (1,2,3,4,5,6) ])
gap> f:=MappingByFunction(X,Y,x->x^2);
MappingByFunction( Group([ (1,2,3) ]), Group(
[ (1,2,3,4) ]), function( x ) ... end )
gap> g:=MappingByFunction(Y,Z,x->x^3);
MappingByFunction( Group([ (1,2,3,4) ]), Group(
```

```
[ (1,2,3,4,5) ]), function( x ) ... end )
gap> h:=MappingByFunction(Z,T,x->x^4);
MappingByFunction( Group([ (1,2,3,4,5) ]),
Group([ (1,2,3,4,5,6) ]), function( x ) ... end )
gap> CompositionMapping(CompositionMapping(h,g),f);
CompositionMapping( CompositionMapping( MappingByFunction(
Group([ (1,2,3,4,5) ]), Group([ (1,2,3,4,5,6) ]),
function( x ) ... end ), MappingByFunction( Group([ (1,2,3,4)
]), Group([ (1,2,3,4,5) ]), function( x ) ... end ) ),
MappingByFunction( Group([ (1,2,3) ]), Group([ (1,2,3,4) ]),
function( x ) ... end ) )
gap> CompositionMapping(CompositionMapping(h,g),f)=
CompositionMapping(h,CompositionMapping(g,f));
true
```

Tanım : Bir $f : X \to Y$ fonksiyonu verilsin. $f \circ g = Id_Y$ ve $g \circ f = Id_X$ eşitliğini sağlayan bir $g : Y \to X$ fonksiyonu varsa, g fonksiyonuna f nin ters fonksiyonu ve benzer şekilde f fonksiyonuna da g nin ters fonksiyonu denir [19].

GAP programında bir fonksiyonun tersi `InverseGeneralMapping()` komutu ile bulunur. `()` işaretleri arasına tersi bulunacak olan fonksiyon yazılmalıdır. Bulunan tersin bir fonksiyon olup olmadığı `IsMapping()` komutu kullanılarak test edilir [17].

Teorem : Bir $f : X \to Y$ fonksiyonunun tersinin varolması için gerekli ve yeter şart f fonksiyonunun birebir ve örten olmasıdır [19].

```
gap> M:=Group((1,2,3,4),(1,2));
Group([ (1,2,3,4), (1,2) ])
gap> N:=Group((1,4,3,2),(1,2));
Group([ (1,4,3,2), (1,2) ])
gap> f:=MappingByFunction(M,N,x->x);
MappingByFunction(Group([(1,2,3,4), (1,2) ]),
Group([ (1,4,3,2), (1,2) ]), function( x ) ... end )
gap> IsBijective(f);
true
```

```
gap> IsInjective(f);
true
gap> IsSurjective(f);
true
gap> tf:=InverseGeneralMapping(f);
InverseGeneralMapping(  MappingByFunction(  Group([  (1,2,3,4),
(1,2) ]), Group([ (1,4,3,2), (1,2) ]), function( x ) ... end ) )
gap> IsMapping(f);
true
gap> CompositionMapping(f,tf)=IdentityMapping(N);
true
gap> CompositionMapping(tf,f)=IdentityMapping(M);
true
```

7 TAMSAYILAR

7.1 Giriş

Bu bölümde tamsayılar da kullanılan kalanlı bölme özellikleri, Euclid algoritması, en büyük ortak bölen, en küçük ortak bölen ve özellikleri hem matematiksel olarak hem de GAP uygulamaları ile birlikte verilmiştir. Matematiksel ifadeler genellikle [7] den alınmıştır.

7.2 OBEB – OKEK

Tanım : $n = qm + r$, $0 \le r < |m|$ ise q ya bölüm, r ye de kalan denir. Verilen m ve n tam sayıları için q ve r yi bulmaya da n yi m ile kalanlı bölme denir [7].

Tanım : m ve n sıfırdan farklı birer tam sayı olsunlar. $d \mid m$ ve $d \mid n$ olacak şekilde $d > 0$ tam sayısı varsa d ye m ile n nin bir ortak böleni denir [7].

Tanım : m ve n sıfırdan farklı birer tam sayı olsunlar. d, m ile n nin ortak böleni olsun. Eğer m ile n nin her e ortak böleni için $e \mid d$ ise d ye m ile n nin en büyük ortak böleni denir ve $(m,n) = d$ ile gösterilir [7].

Tanım : m ve n sıfırdan farklı bir tam sayı olsunlar. İki tam sayının en büyük ortak böleni 1 ise bu iki sayıya aralarında asal sayılar denir [6].

$n = qm + r$, $0 \le r < |m|$ ise q ya bölüm, r ye de kalan denir. Verilen m ve n tam sayıları için q ve r yi bulmaya da n yi m ile kalanlı bölme denir. a ile b nin iki en büyük ortak böleni d_1 ve d_2 ise $d_1 \mid d_2$ ve $d_2 \mid d_1$ ve $d_1 = d_2$ olacağı anlaşılır. Yani iki tamsayının en büyük ortak böleni tektir [7].

Teorem : Sıfırdan farklı herhangi iki sayının en büyük ortak böleni vardır ve $d = (m,n)$ ise $d = xm + yn$ olacak şekilde $\exists x, y \in \mathbb{Z}$ bulunabilir [7].

İspat : $S = \{xm + yn : xm + yn > 0,\ x, y \in \mathbb{Z}\}$ diyelim. $x = m$ ve $y = n$ için $m^2 + n^2 \in S$ ve $S \neq \varnothing$ olur. Pozitif tam sayıların iyi sıralı olma özelliğinden, S nin bir d gibi en küçük elemanı vardır. Şimdi $d = (m,n)$ olduğunu gösterelim.

m yi d ile kalanlı olarak bölelim. $m = qd + r$ ve $0 \le r < d$ olacak şekilde $\exists q, r \in \mathbb{Z}$ bulunabilir. $d = xm + yn$ şeklinde olduğundan,

$$m = q(xm + yn) + r \Rightarrow r = m(1 - qx) + n(-qy)$$

dir. Bu durumda $r \neq 0$ ise $r \in S$ olur ki, $r < d$ olduğu için $d \in S$ nin en küçük oluşu ile çelişir. Şu halde $r = 0$, yani $d \mid m$ olmalıdır.

m ve n simetrik rol oynadıkları için, benzer şekilde $d \mid n$ olduğu da gösterilebilir. Böylece d nin m ile n nin bir ortak böleni olduğu anlaşılır.

$e \in \mathbb{Z}$, $e \mid m$ ve $e \mid n$ olsun. $m = eu$ ve $n = ev$ olacak şekilde, $\exists u, v \in \mathbb{Z}$ bulunabilir.

$$d = xm + yn = x(eu) + y(ev) = e(xu + yv),$$

olduğundan $e \mid d$ dir. Sonuç olarak $d = (m, n)$ ve $d \in S$ olduğundan $\exists x, y \in Z$ için $d = xm + yn$ olur.

Şimdi iki tam sayının en küçük ortak bölenlerini bulmak için bir yöntem inceleyelim. m ve n pozitif tam sayılar olsun. Ard arda aşağıdaki kalanlı bölmeleri yapalım:

$$\begin{aligned}
n &= q_1 m + r_1 && \text{ve} \quad 0 \le r_1 < m && (1)\\
m &= q_2 r_1 + r_2 && \text{ve} \quad 0 \le r_2 < r_1 && (2)\\
r_1 &= q_3 r_2 + r_3 && \text{ve} \quad 0 \le r_3 < r_2 && (3)\\
\cdots &= \cdots\cdots && \cdots\cdots && \cdots\\
r_{k-1} &= q_{k+1} r_k + r_{k+1} && \text{ve} \quad 0 \le r_{k+1} < r_k && (k+1)\\
r_k &= q_{k+2} r_{k+1} + 0 && \cdots\cdots && (k+2)
\end{aligned}$$

şeklinde kalan 0 oluncaya kadar devam edelim. Kalanlar gittikçe küçüldüğünde, $m > r_1 > r_2 > \cdots$ olduğuna dikkat edersek, sonlu bir adım sonra 0 kalanına ulaşılacağı açıktır [7].

Teorem (Euclid Algoritması) : Yukarıda ard arda yapılan kalanlı bölmeler arasında sıfırdan farklı en son kalan m ile n nin en büyük ortak bölenidir, yani $r_{k+1} = (m, n)$ dir [7].

İspat : Önce r_k nın m ile n nin bir ortak böleni olduğunu gösterelim. Kalanlı bölmelerde son eşitlikten, $r_{k+1} \mid r_k$ ve sondan bir önceki eşitlikten,

$$r_{k+1} \mid q_{k+1} r_k + r_{k+1} = r_{k-1}$$

ve bu şekilde devam edilerek ;

$$r_{k+1} \mid r_k, r_{k-1} \ldots, r_1, m, n$$

elde edilir. Şimdi $e \in \mathbb{Z}$, $e \mid m$ ve $e \mid n$ ise $e \mid r_{k+1}$ olacağını gösterelim.

$$e \mid n \text{ ve } e \mid m \Rightarrow e \mid n - q_1 m = r_1,$$
$$e \mid m \text{ ve } e \mid r_1 \Rightarrow e \mid m - q_2 r_1 = r_2,$$
$$e \mid r_1 \text{ ve } e \mid r_2 \Rightarrow e \mid r_1 - q_3 r_2 = r_3$$

ve böyle devam ederek, $e \mid r_{k+1}$ bulunur [7].

Örnek : 24 ve 550 sayılarının en büyük ortak bölenini bulalım [20].

$$550 = 24.22 + 22$$
$$24 = 22.1 + 2$$
$$22 = 11.2 + 0$$

olduğundan yukarıdaki teoreme göre $(550, 24) = 2$ bulunur.

Örnek : 72139 ve 3146 sayılarının en büyük ortak bölenini bulalım [6].

$$72139 = 22.3146 + 2927$$
$$3146 = 1.2927 + 219$$
$$2927 = 13.219 + 80$$
$$219 = 2.80 + 59$$
$$80 = 1.59 + 21$$
$$59 = 2.21 + 17$$
$$21 = 1.17 + 4$$
$$17 = 4.4 + 1$$
$$4 = 4.1 + 0$$

olduğundan $(72139, 3146) = 1$ bulunur.

Şimdi bu örnekleri GAP programı kullanarak çoğaltalım. GAP programı birçok fonksiyonu bünyesinde barındırır. GAP programı içerisindeki bu fonksiyonlardan birçoğu büyük harflerle başlarlar. `Gcd()` fonksiyonu sıfırdan farklı değerlerin en büyük ortan bölenini (OBEB) hesaplar [17].

```
gap> Gcd(171,30);
3
gap> Gcd(72139,3146);
1
gap> Gcd(1246789,1896);
```

```
1
gap> Gcd(122,244);
122
gap> Gcd(122,64,18);
2
gap> Gcd(1,2,3,4,5);
1
```

Euclid Algoritması ile en büyük ortak bölen bulunduktan sonra, $d=(m,n)$ ise $d=xm+yn$ olacak şekilde $x,y\in Z$ tam sayıları da bulunabilir. Gerçekten Euclid Algoritmasındaki

1. bölmeden $r_1=n-q_1m$

2. bölmeden $r_2=m-q_2r=m-q_2(n-q_1m)=-q_2n+(1-q_2q_1)m$

bulunur. Böyle devam edilirse, sıfırdan farklı son kalan yani $r_{k+1}=(m,n)=xm+yn$ şeklinde yazılmış olur. Aynı şey, r_{k+1} in değeri, bir önceki eşitlikte yerine koyarak birinci eşitliğe kadar devam edilerek de elde edilebilir.

Örnek : $(171,30)=3=x.171+y.30$ olacak şekilde $\exists x,y\in\mathbb{Z}$ bulalım [6].

$$\begin{aligned}3&=21-2.9\\&=21-2.(30-21)\\&=3.21+2.30\\&=3.(171-5.30)-2.30\\&=3.171-17.30\end{aligned}$$

bulunur. $x=3$ ve $y=-17$ alınabilir.

Örnek : $(72139,3146)=1=x.72139+y.3146$ olacak şekilde $\exists x,y\in\mathbb{Z}$ bulalım [6].

$$\begin{aligned}1&=17-4.4=17-4.(21-17)\\&=5.17-4.21=5.(59-2.21)-4.21\\&=5.59-14.21=5.59-14.(80-59)\\&=19.59-14.80=19.(219-2.80)-14.80\\&=19.219-52.80=19.219-52.(2927-13.219)\\&=695.219-52.2927=695.(3146-2927)-52.2927\\&=695.3146-747.2927=695.3146-747.(72.139-22.3146)\\&=-747.72139+17129.3146\end{aligned}$$

bulunur. $x=-747$ ve $y=17129$ alınabilir.

Şimdi bu örneklerimizi GAP programında inceleyelim. `Gcdex()` fonksiyonu `Gcd()` fonksiyonu ile en büyük ortak bölenini bulduğumuz a ve b sayılarının bir lineer kombinasyon olarak yazılmasını sağlar.

```
gap> Gcdex(171,30);
rec( gcd := 3, coeff1 := 3, coeff2 := -17, coeff3 := -10, coeff4
:= 57 )
gap> Gcdex(72139,3146);
rec( gcd := 1, coeff1 := -747, coeff2 := 17129, coeff3 := 3146,
  coeff4 := -72139 )
gap> Gcdex(1246789,1896);
rec( gcd := 1, coeff1 := 589, coeff2 := -387320, coeff3 := -
1896,
  coeff4 := 1246789 )
```

`Gcdex()` fonksiyonunu kullanıldığın da $Gcd(m,n)=xm+yn$ olmak üzere `coeff1` sayısı x değerini `coeff2` sayısı y değerini ifade etmektedir. Burada ki `coeff3` ve `coeff4` sayıları ise $0=zm+tn$ olmak üzere z ve t sayılarını ifade eder [16].

Yukarıdaki örnekte $z=3146$ ve $t=-72139$ olup $m=72139$, $n=3146$ değerleri için

$$zm+tn=3146\cdot 72139+(-72139\cdot 3146)=0$$

eşitliği sağlanır.

Dikkat edilmesi gereken bir diğer nokta ise `Gcdex()` fonksiyonunun iki değer için incelenebildiğidir. `Gcdex(10,20,30);` biçimindeki bir komut dizisi GAP programının hata vermesine yol açacaktır.

```
gap> Gcdex(10,20,30);
Function: number of arguments must be 2 (not 3)
not in any function
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
```

```
you can replace the argument list <args> via 'return <args>;' to
continue
brk>
```

Tanım : m ve n sıfırdan farklı iki tam sayı olsun. $m \mid k$ ve $n \mid k$ olacak şekilde $k > 0$ tam sayısı varsa k ya m ile n nin bir ortak katı denir [7].

Tanım : m ve n sıfırdan farklı iki tam sayı olsun. k, m ile n nin bir katı olsun. Eğer m ile n her ortak katı için $k \mid t$ ise k ya m ile n nin en küçük ortak katı denir ve $k = [m,n]$ ile gösterilir [7].

GAP programında `Lcm()` fonksiyonu sıfırdan farklı değerlerin ortak katlarının en küçüğünü hesaplar [17].

```
gap> Lcm(122,244);
244
gap> Lcm(48,16,8);
48
```

Teorem : a ve b pozitif tam sayılar olsun.

$$(a,b)[a,b] = ab$$

eşitliği sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> a:=25;
25
gap> b:=12;
12
gap> Gcd(a,b);
1
gap> a*b=Gcd(a,b)*Lcm(a,b);
true
```

Tanım : Pozitif n tamsayısı için $1 \le a < n$ ve $(a,n)=1$ olan a tam sayılarının sayısı $\varphi(n)$ ile gösterilir ve Euler φ fonksiyonu denir. Euler φ fonksiyonu,

i) p asal ise $\varphi(n) = p-1 = p(1-1/p)$

ii) p asal ve $a \in \mathbb{N}$ ise $\varphi(p^a) = p^a - p^{a-1} = p^a(1-1/p)$

iii) $(m,n)=1$ ise $\varphi(mn) = \varphi(m)\varphi(n)$

iv) $m = p_1^{a_1} p_2^{a_2} \dots p_r^{a_r}$ ise $\varphi(m) = \varphi(p_1^{a_1})\varphi(p_2^{a_2}) \dots \varphi(p_r^{a_r})$

özelliklerini sağlar [7].

Örnek : Aşağıdaki örnekler Euler φ fonksiyonunun özelliklerini göstermektedir [7].

$$\varphi(3^5) = 3^5.(1-1/3) = 3^4.2 = 162,$$
$$\varphi(2^4.3^2) = \varphi(2^4).\varphi(3^2) = 2^3.3.2 = 48,$$
$$\varphi(50) = \varphi(2).\varphi(5^2) = 1.5.4 = 20,$$
$$\varphi(63) = \varphi(3^2).\varphi(7) = 6.6 = 36,$$
$$\varphi(100) = \varphi(2^2).\varphi(5^2) = 2.5.4 = 40.$$
$$\varphi(2) = 2-1 = 1 \text{ dir.}$$

GAP programında `Phi()` fonksiyonu Euler φ fonksiyonunun değerini hesaplamakta kullanılır [17].

```
gap> Phi(3^5);
162
gap> Phi((2^4)*(3^2));
48
gap> Phi(50);
20
gap> Phi(63);
36
gap> Phi(100);
40
gap> Phi(561);
320
gap> Phi(2);
1
```

Teorem : n bir doğal sayı olmak üzere ;

$$n \text{ tek ise } \varphi(2.n) = 2\varphi(n)$$

$$n \text{ çift ise } \varphi(2.n) = \varphi(n)$$

eşitlikleri sağlanır [7].

Şimdi bu önermenin doğruluğunu GAP programı kullanarak gösterelim.

```
gap> n:=3;
3
gap> Phi(2*n)=Phi(n);
true
gap> n:=1234345;
1234345
gap> Phi(2*n)=Phi(n);
true
gap> n:=12;
12
gap> Phi(2*n)=2*Phi(n);
true
gap> n:=125468;
125468
gap> Phi(2*n)=2*Phi(n);
true
gap> n:=5;
5
gap> Phi(2*n)=2*Phi(n);
false
gap> n:=4;
4
gap> Phi(2*n)=Phi(n);
false
```

8 MODÜLER ARİTMETİK

Bu bölümde modüler aritmetik ile ilgili özellikler, tanımlar ve teoremler GAP örnekleri ile birlikte verilmiştir.

Tanım : m sıfırdan farklı bir tamsayı olsun. $a,b \in \mathbb{Z}$ için ;

$$a \equiv b(\text{mod}\, m) \Rightarrow m \mid a-b$$

ile tanımlanır ve a ile b mod m denktirler denir [7].

$m \mid a-b \Rightarrow -m \mid a-b$ olduğundan, $m > 0$ kabul edilebilir.

Teorem : Yukarıda tanımlanan $\equiv$ bağıntısı $\mathbb{Z}$ de bir denklik bağıntısıdır [7].

İspat :

i) Yansıma : $\forall a \in \mathbb{Z}$ için; $m \mid 0 = a-a$ olduğundan $a \equiv a(\text{mod}\, m)$ dir.

ii) Simetri : $a,b \in \mathbb{Z}$ için; $a \equiv b(\text{mod}\, m)$ olsun. Şu halde $m \mid a-b$ ve $m \setminus b-a = -(a-b)$ olduğundan, $b \equiv a(\text{mod}\, m)$ dir.

iii) Geçişme : $a,b,c \in \mathbb{Z}$ için , $a \equiv b(\text{mod}\, m)$ ve $b \equiv c(\text{mod}\, m)$ olsun.

$m \mid a-b$ ve $m \mid b-c \Rightarrow m \mid a-c = (a-b)+(b-c)$ olduğundan, $a \equiv c(\text{mod}\, m)$ dir [7].

Tanım : $\mathbb{Z}$ deki $\equiv$ denklik sınıflarının belirttiği denklik sınıflarına, m modülüne göre $(\text{mod}\, m)$ **kalan sınıfları** denir ve tüm kalan sınıfları kümesi $\mathbb{Z}_m$ ile gösterilir. $k \in \mathbb{Z}$ nin denklik sınıfı, $\overline{k} = \{x \in \mathbb{Z} : m \mid k-x\}$ dir [7].

Örnek : $25 \equiv 4(\text{mod}\, 7)$ ve $33 \equiv 2(\text{mod}\, 3)$ olduğundan $25 \in \overline{4}$ ve $33 \in \overline{2}$ dir.

$\mathbb{Z}_{10} = \{\overline{0},\overline{1},\overline{2},\overline{3},\overline{4},\overline{5},\overline{6},\overline{7},\overline{8},\overline{9}\}$ dir.

Şimdi GAP programın kullanarak mod hesaplaması yapalım;

```
gap> c:=25 mod 7;
4
gap> d:=12345 mod 11;
3
```

GAP programında mod hesabı yapılırken karakterler arasında boşluk bırakmak önemlidir. Boşluk bırakılmadığı takdirde GAP programı, komutu bir tanımlayıcı ismi olarak görecektir [17].

```
gap> 25mod7;
Variable: '25mod7' must have a value
```

Teorem : $a \equiv a_1 (\mathrm{mod}\, m)$ ve $b \equiv b_1 (\mathrm{mod}\, m)$ ise

$$a+b \equiv a_1 + b_1 (\mathrm{mod}\, m)$$

eşitliği sağlanır [7].

İspat :

$$a \equiv a_1 (\mathrm{mod}\, m) \text{ ve } b \equiv b_1 (\mathrm{mod}\, m) \Rightarrow m \mid a-a_1, m \mid b-b_1$$
$$\Rightarrow m \mid (a+b)-(a_1+b_1)$$
$$\Rightarrow a+b \equiv a_1 + b_1 (\mathrm{mod}\, m)$$

elde edilir [7].

Şimdi bu teoremi GAP programı kullanarak inceleyelim.

```
gap> a:=45;
45
gap> b:=52;
52
gap> m:=7;
7
gap> a1:=a mod m;
3
gap> b1:=b mod m;
3
gap> c:=a1+b1; d:=(a+b) mod m; a1+b1=(a+b) mod m;
6
6
true
```

Tanım : $\overline{a}, \overline{b} \in \mathbb{Z}_m$ için $\overline{a} \oplus \overline{b} \in \overline{a+b}$ ile tanımlanır [7].

Teorem : $a \equiv a_1 (\mathrm{mod}\, m)$ ve $b \equiv b_1 (\mathrm{mod}\, m)$ ise

$$a.b \equiv a_1.b_1 (\mathrm{mod}\, m)$$

eşitliği sağlanır [7].

İspat :

$$\begin{aligned} a \equiv a_1 (\mathrm{mod}\, m) \text{ ve } b \equiv b_1 (\mathrm{mod}\, m) &\Rightarrow m \mid a - a_1, m \mid b - b_1 \\ &\Rightarrow m \mid a_1.(b - b_1) + b.(a - a_1) \\ &\Rightarrow a.b \equiv a_1.b_1 (\mathrm{mod}\, m) \end{aligned}$$

elde edilir [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> a:=452;
452
gap> b:=157;
157
gap> m:=11;
11
gap> a1:=a mod m;
1
gap> b1:=b mod m;
3
gap> c:=a1*b1; d:=(a*b) mod m; a1*b1=(a*b) mod m;
3
3
true
```

Tanım : $a \equiv a_1 (\mathrm{mod}\, m)$ ve $b \equiv b_1 (\mathrm{mod}\, m)$ ise

$$a.b \equiv a_1.b_1 (\mathrm{mod}\, m)$$

eşitliği sağlanır [7].

Teorem : $\mathbb{Z}_m$ de $\forall \bar{a}, \bar{b}, \bar{c} \in \mathbb{Z}_m$ olmak üzere $\oplus$ işlemi için,

i) $\bar{a} \oplus \bar{b} = \bar{b} \oplus \bar{a}$ dir. (Değişme Özelliği)

ii) $\bar{a} \oplus (\bar{b} \oplus \bar{c}) = (\bar{a} \oplus \bar{b}) \oplus \bar{c}$ dir. (Birleşme Özelliği)

iii) $\overline{a} \oplus \overline{0} = \overline{a}$ dır. ($\overline{0}$ etkisiz eleman)

iv) $\overline{a} \oplus \overline{x} = \overline{0}$ olacak şekilde $\exists \overline{x} \in \mathbb{Z}_m$ bulunabilir. (Ters Eleman)

özellikleri sağlanır [7].

Şimdi bu önermeyi GAP programı kullanarak inceleyelim.

```
gap> a:=452; b:=865; c:=51; m:=43;
452
865
51
43
gap> a1:=a mod m; b1:=b mod m; c1:=c mod m;
22
5
8
gap> ma:=a1 mod m;
22
gap> mb:=b1 mod m;
5
gap> ma+mb=mb+ma;
true
gap> mc:=c1 mod m;
8
gap> ma+(mb+mc)=(ma+mb)+mc;
true
gap> m0:=0 mod m;
0
gap> ma+m0=ma;
true
gap> x1:=-a1;
-22
gap> mx:=x1 mod m;
21
gap> (ma+mx) mod m=m0;
true
```

Teorem : $\mathbb{Z}_m$ de $\forall \overline{a},\overline{b},\overline{c} \in \mathbb{Z}_m$ olmak üzere $\odot$ işleminin için,

i) $\overline{a} \odot \overline{b} = \overline{b} \odot \overline{a}$ dir. (Değişme Özelliği)

ii) $\overline{a} \odot (\overline{b} \odot \overline{c}) = (\overline{a} \odot \overline{b}) \odot \overline{c}$ dir. (Birleşme Özelliği)

iii) $\overline{a} \odot \overline{1} = \overline{a}$ dır. ($\overline{1}$ etkisiz eleman)

iv) $\overline{a} \oplus \overline{0} = \overline{0}$ dır. ($\overline{0}$ yutan eleman)

özellikleri sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> ma*mb=mb*ma;
true
gap> ma* (mb*mc) =ma* (mb*mc) ;
true
gap> m1:=1 mod m;
1
gap> ma*m1=ma;
true
gap> ma*m0=m0;
true
```

Teorem : $\forall \overline{a},\overline{b},\overline{c} \in \mathbb{Z}_m$ için,

$$\overline{a} \odot (\overline{b} \oplus \overline{c}) = (\overline{a} \odot \overline{b}) \oplus (\overline{a} \odot \overline{c})$$

eşitliği sağlanır. ($\odot$ nın $\oplus$ üzerine dağılma özelliği) [7]

İspat :

$$\overline{a} \odot (\overline{b} \oplus \overline{c}) = \overline{a} \oplus \overline{b+c}) = \overline{a(b+c)}$$

ve

$$(\overline{a} \odot \overline{b}) \oplus (\overline{a} \odot \overline{c}) = \overline{ab} \oplus \overline{ac} = \overline{ab+ac}) \text{ dir.}$$

$\mathbb{Z}$ de, çarpmanın toplama üzerine dağılma özelliğinden istenen eşitlik elde edilir [7].

Bu teoremin doğruluğunu GAP programı kullanarak inceleyelim.

```
gap> ma* (mb+mc) = (ma*mb) + (ma*mc) ;
true
```

Tanım : $\mathbb{Z}_m$ de kendileri $\bar{0}$ dan farklı olduğu halde çarpımları $\bar{0}$ olan sınıflara **sıfır bölen sınıflar** denir [7].

Örnek :

$\mathbb{Z}_{12}$ de sıfır bölenler $\bar{2},\bar{3},\bar{4},\bar{6},\bar{8}$ dir.

$\mathbb{Z}_8$ de sıfır bölenler $\bar{2},\bar{4},\bar{6}$ dir.

Bu örnekleri GAP programı kullanarak inceleyelim.

```
gap> m2:=2 mod 12; m6:=6 mod 12;
2
6
gap> (m2*m6) mod 12=m0;
true
gap> m4:=4 mod 8; m6:=6 mod 8;
4
6
gap> (m4*m6) mod 8=m0;
true
```

Teorem : $a \equiv b(\text{mod}\, m) \Rightarrow (a,m) = (b,m)$ eşitliği sağlanır [7].

İspat :

$a \equiv b(\text{mod}\, m) \Rightarrow m \mid a-b \Rightarrow a = b+mk,\ \exists k \in \mathbb{Z}$ dır. $(a,m) = d$ olsun. $d \mid a$ ve $d \mid m$ olduğundan $d \mid b = a-mk$ dır. Şu halde d , b ile m nin bir ortak bölenidir. Eğer t herhangi bir ortak bölen ise $t \mid b$ ve $t \mid m$ ise $t \mid a = b+mk$ olacağından, $t \mid d = (a,m)$ bulunur. Bu durumda $d = (b,m)$ dir [7].

Şimdi bu teoremi GAP programı kullanarak inceleyelim.

```
gap> a:=753; m:=12 ;
753
12
gap> b:= a mod m;
9
```

```
gap> Gcd(a,m); Gcd(b,m); Gcd(a,m)=Gcd(b,m);
3
3
true
```

Tanım : $\bar{a} \in \mathbb{Z}$ sınıfı için, $(a,m)=1$ ise $\bar{a}$ sınıfına bir **asal kalan sınıfı** denir. $\mathbb{Z}_m$ in bütün asal kalan sınıfları $\mathbb{Z}_m^*$ ile gösterilir [7].

m modülü asal ise $\mathbb{Z}_m^* = \mathbb{Z}_m - \{\bar{0}\}$ dir. Burada dikkat edilecek başka bir nokta $\mathbb{Z}_m^*$ nin eleman sayısının $\varphi(m)$ Euler φ fonksiyonu ile hesaplanacağıdır.

Şimdi GAP programı kullanarak aralarında asal olan sayıları hesaplayan bir fonksiyon oluşturalım. Bu fonksiyon için bir metin editörü kullanarak `arasal.gi` isimli bir dosya oluşturalım. Fonksiyon girilen değerden daha küçük olan ve girilen değerle aralarında asal olan tüm sayıları verecektir.

```
arasal:=function(n)
local s,i,o;
o:=One(Integers mod n);
s:=n -> Filtered([1..n-1], i -> Gcd(i,n)=1);
return s(n)*o;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `arasal.gi` dosyasını GAP programı aracılığı ile çalıştıralım ve içerisinde yer alan `arasal` isimli fonksiyonu kullanarak herhangi bir n sayısından küçük ve n sayısı ile aralarında asal olan sayıları hesaplayalım.

```
gap> Read("arasal.gi");
gap> arasal(20);
[ ZmodnZObj( 1, 20 ), ZmodnZObj( 3, 20 ), ZmodnZObj( 7, 20 ),
  ZmodnZObj( 9, 20 ), ZmodnZObj( 11, 20 ), ZmodnZObj( 13, 20 ),
  ZmodnZObj( 17, 20 ), ZmodnZObj( 19, 20 ) ]
```

Fonksiyon 20 sayısıyla aralarında asal olan 20 den küçük sayıları bularak ekrana görüntüledi.

Gerçekte GAP programı içerisinde aralarında asal olan sayıları hesaplayan temel bir komut zaten vardır. Bu komut `PrimeResidues()` komutudur. Bu fonksiyon () işaretleri arasına yazılan sayıdan küçük olan ve bu sayıyla aralarında asal olan sayıları bir liste halinde ekrana yazdırır [17].

```
gap> PrimeResidues(20);
[ 1, 3, 7, 9, 11, 13, 17, 19 ]
gap> PrimeResidues(100);
[ 1, 3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39,
41, 43, 47, 49,51, 53, 57, 59, 61, 63, 67, 69, 71, 73, 77, 79,
81, 83, 87, 89, 91, 93, 97, 99 ]
```

Teorem : İki asal kalan sınıfının çarpımı da bir asal kalan sınıfıdır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> a:=12; b:=11; m:=7;
12
11
7
gap> Gcd(a,m); Gcd(b,m);
1
1
gap> a1:=a mod m;; b1:=b mod m;;
gap> ma:=a1 mod m; mb:=b1 mod m;
5
4
gap> Gcd((ma*mb) mod m,m);
1
```

9 GRUP TEORİ

9.1 Giriş

Bu bölümde matematikte bilinen grup notasyonlarının yanı sıra GAP programında var olan notasyonlar da sunulacaktır. Bu notasyonlar bilgisayarda grup oluşturmada ve grupla ilgili işlemlerin yapılmasına yardımcı olacaktır. Her GAP programı fonksiyonun kullanımı için basit örnekler verilmiş olup bu örnekler GAP programında grupların kullanılmasını kolaylaştırıcı örneklerdir.

Gruplarda kullanılan üreteçler 30. dereceden gruplar içinde liste halinde bu çalışma sonunda verilmiştir. Bu bölümde kullanılan GAP programı fonksiyonları, GAP programı içerisinde yapmış oldukları işlevlerle açıklanmışlardır. Bu fonksiyonlar , `Group()`, `GroupWithGenerators()`, `IsGroup()`, `IsAbelian()`, `Size()`, `Elements()`, `Identity()`, `Inverse()`, `DirectProduct()`, `SymmetricGroup()`, `DihedralGroup()`, `GeneratorsOfGroup()` fonksiyonlarıdır. Bu fonksiyonlarla grup teoride kullanılan grup aksiyomlarındaki notasyonlar da GAP programı fonksiyonları ile açıklanmıştır.

9.2 Grup Kavramı

Tanım : G boş olmayan bir küme ve $o: G\times G \to G$ ikili işlemi verilsin;

G1) Her $x, y, z \in G$ için

$$(xoy)oz = xo(yoz)$$

G2) Her $x \in G$ için

$$eox = xoe = x$$

olacak şekilde en az bir $e \in G$ vardır.

G3) Her $x \in G$ için

$$x^{-1}ox = xox^{-1} = e$$

olacak şekilde en az bir $x^{-1} \in G$ elemanı vardır.

Bu üç özellik sağlanıyorsa (G,o) ikilisine bir grup denir.

Buradaki e elemanına G nin o işlemine göre birim (veya etkisiz) elemanı ve x^{-1} elemanına G nin o işlemine göre ters (veya tersinir) elemanıdır denir [3].

GAP programında gruplar `Group()` ve `GroupWithGenerators()` fonksiyonları ile oluşturulurlar. Bu fonksiyonlar temelde aynı işlem yapmalarına rağmen aralarında iki temel fark vardır. Bu farklardan bir tanesi bu fonksiyonların kullanımı ile ilgilidir. `Group()` fonksiyonu kullanılırken `()` işaretleri arasına oluşturulacak olan grubun üreteçleri yazılmalıdır. `GroupWithGenerators()` fonksiyonu kullanılırken `()` işaretleri arasına yine grubun üreteçleri yazılır ancak bu üreteçler liste halinde verilmelidir [17].

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> h:=Group((1,2),(3,4));
Group([ (1,2), (3,4) ])
gap> k:=GroupWithGenerators([(1,2),(3,4)]);
Group([ (1,2), (3,4) ])
```

Bu iki fonksiyon arasındaki diğer önemli fark ise `Group()` fonksiyonunda `()` işaretleri arasına yazılan üreteçler ile oluşturulan grubun gerçek üreteçleri arasında fark olabilir. `GroupWithGenerators()` fonksiyonunda ise `()` işaretleri arasına yazılan üreteçler ile grubun gerçek üreteçleri aynı olmak zorundadır.

Bir cebirsel yapının grup olup olmadığını test edebilmek için GAP programında `IsGroup()` komutu kullanılır [17].

```
gap> IsGroup(G);
true
```

Eğer bir grup da,

G4) Her $x, y \in G$ için

$$xoy = yox$$

eşitliği sağlanıyorsa bu gruba değişmeli grup veya abelyen grup denir [3].

GAP programında bir grubun değişmeli olup olmadığını test edebilmek için `IsAbelian()` fonksiyonu kullanılır. Bu fonksiyon içerisindeki `()` işaretleri arasına değişmeli olup olmadığı test edilecek olan grup yazılmalıdır [17].

```
gap> IsAbelian(G);
true
```

G kümesi sonlu ise G grubuna sonlu grup denir. Aksi halde grup sonsuzdur denir. G nin eleman sayısına grubun mertebesi denir ve $|G|$ ile gösterilir. Eğer grup sonsuz ise sonsuz mertebeden denir [3].

GAP programında `Size()` ve `Order()` fonksiyonları bir grubun mertebesini bulmak için kullanılır. Bu fonksiyonlar içerisindeki `()` işaretleri arasına mertebesi bulunacak grup yazılmalıdır [17].

```
gap> Size(G); Order(G);
4
4
```

GAP programında bir grubun elemanlarının bulunabilmesi için `Elements()` fonksiyonu kullanılır. Buradaki `()` işaretleri arasına elemanları bulunacak olan grup yazılmalıdır [17].

```
gap> Elements(G);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
```

Grup elemanları içerisinde listelenen ilk eleman her zaman birim elemanı gösterir.

```
gap> Eleman:=Elements(G);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> Birim:=Eleman[1];
()
```

Grup elemanlarını bulmadan da `Identity()` fonksiyonunu kullanarak bir grubun birim elemanı bulunabilir. Buradaki `()` işaretleri arasına birim elemanı bulunacak grup yazılmalıdır [17].

```
gap> Identity(G);
()
```

Şimdi grup aksiyomlarını GAP programını kullanarak inceleyelim. $a,b,c \in G$ olmak üzere a,b ve c elemanları alalım ve bu elemanların birleşme özelliğini sağladığını gösterelim.

```
gap> a:=Eleman[2]; b:=Eleman[3]; c:=Eleman[4];
(1,2,3,4)
(1,3)(2,4)
(1,4,3,2)
gap> a*(b*c); (a*b)*c; a*(b*c)=(a*b)*c;
(1,3)(2,4)
(1,3)(2,4)
true
```

Gruptaki bir eleman ile birim elemanın çarpımı gene o elemanı vermelidir.

```
gap> a*Birim; Birim*a; a*Birim=a; Birim*a=a;
(1,2,3,4)
(1,2,3,4)
true
true
```

Teorem : G bir grup olmak üzere birim eleman bir tektir [3].

Gruptaki bir elemanın tersini bulabilmek için GAP programında `Inverse()` fonksiyonu yada `^-1` ifadesi kullanılır. Gruptaki bir elemanla tersinin çarpımı birim elemanı vermek zorundadır. Yukarıdaki teoremin ispatı ve `Inverse()` fonksiyonunun kullanımı aşağıdaki gibi incelenebilir [17].

```
gap> a;
(1,2,3,4)
gap> at:=a^-1;
(1,4,3,2)
gap> at:=Inverse(a);
(1,4,3,2)
gap> a*at; at*a; a*at=Birim; at*a=Birim;
()
()
true
true
```

Teorem : G bir grup olmak üzere her elemanın tersi bir tektir [3].

Teorem : G bir grup olmak üzere $x, y \in G$ için

$$xy = e \Rightarrow y = x^{-1} \text{ ve } x = y^{-1}$$

eşitlikleri sağlanır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> x:=Eleman[2];
(1,2,3,4)
gap> y:=Eleman[4];
(1,4,3,2)
gap> x*y;
()
gap> x^-1=y; Inverse(x)=y;
true
true
gap> y^-1=x; Inverse(y)=x;
true
true
```

Teorem : G bir grup olmak üzere $x, y \in G$ için

$$(xy)^{-1} = y^{-1}x^{-1}$$

eşitliği sağlanır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> x:=Eleman[3];
(1,3)(2,4)
gap> y:=Eleman[4];
(1,4,3,2)
gap> (x*y)^-1=(y^-1)*(x^-1);
true
gap> Inverse(x*y)=Inverse(y)*Inverse(x);
true
```

Teorem : G bir grup $x \in G$ ve $e \in G$ nin birim elemanı olmak üzere

$$(x^{-1})^{-1} = x \text{ ve } e^{-1} = e$$

eşitlikleri sağlanır [11].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> (x^-1)^-1=x; (y^-1)^-1=y;
true
true
gap> Inverse(Inverse(x))=x; Inverse(Inverse(y))=y;
true
true
gap> e:=Identity(G);
()
gap> e^-1=e; Inverse(e)=e;
true
true
```

Eğer bir grup değişmeli ise elemanlarını yerlerinin değiştirildiğinde çarpım sonucu değişmemelidir.

```
gap> a*b=b*a;
true
```

Tanım : G herhangi bir grup olsun. $x \in G$ elemanının kuvvetleri aşağıdaki şekilde tanımlanır.

$$x^0 = e; \quad x^1 = x \text{ ve } 1 \le n \in \mathbb{Z} \text{ için } x^{n+1} = x^n.x$$

ve x in negatif kuvvetleri için $1 \le n \in \mathbb{Z}$ olmak üzere $x^{-n} = (x^n)^{-1}$ olarak tanımlanır [3].

```
gap> x:=Eleman[3];
(1,3)(2,4)
gap> x^0; x^0=Birim;
()
true
gap> x^1; x^1=x;
(1,3)(2,4)
true
```

```
gap> n:=2;
2
gap> x^(n+1); x^n*x; x^(n+1)=x^n*x;
(1,3)(2,4)
(1,3)(2,4)
true
gap> x^-n; (x^n)^-1; x^-n=(x^n)^-1;
()
()
true
gap> n:=4;
4
gap> x^-n; (x^n)^-1; x^-n=(x^n)^-1;
()
()
true
```

Teorem : G bir grup olsun. Bu durumda her $x \in G$, $m,n \in \mathbb{Z}$ olmak üzere

i) $x^m x^n = x^{m+n}$,

ii) $(x^m)^n = x^{mn}$,

iii) G abelyen ise $(xy)^n = x^n.y^n$

eşitlikleri sağlanır [3].

Bu teoremin GAP programı kullanarak inceleyelim.

```
gap> n:=12; m:=5;
12
5
gap> (x^m)*(x^n); x^(m+n); (x^m)*(x^n)=x^(m+n);
(1,3)(2,4)
(1,3)(2,4)
true
gap> (x^m)^n; x^(m*n); (x^m)^n=x^(m*n);
()
()
```

```
true
```

Örnek olarak incelenen grup abelyen bir grup olduğundan yukarıdaki teoremin üçüncü şıkkı da grup için geçerlidir.

```
gap> IsAbelian(G);
true
gap> x:=Eleman[2]; y:=Eleman[4]; n:=7;
(1,2,3,4)
(1,4,3,2)
7
gap> (x*y)^n; (x^n)*(y^n); (x*y)^n=(x^n)*(y^n);
()
()
true
```

Şimdi abelyen olmayan bir grup oluşturalım ve grubun elemanlarının yukarıda ki teoremin üçüncü şıkkını sağlamadığını görelim.

```
gap> K:=Group((1,2,3,4),(1,3));
Group([ (1,2,3,4), (1,3) ])
gap> IsAbelian(K);
false
gap> Elemanlar:=Elements(K);
[(),(2,4),(1,2)(3,4),(1,2,3,4),(1,3),(1,3)(2,4),(1,4,3,2),
(1,4)(2,3)]
gap> x:=Elemanlar[2]; y:=Elemanlar[4]; n:=7;
(2,4)
(1,2,3,4)
7
gap> (x*y)^n; (x^n)*(y^n); (x*y)^n=(x^n)*(y^n);
(1,2)(3,4)
(1,4)(2,3)
false
```

Örnek : Dört elemanlı bir grubun değişmeli grup olduğunu gösterelim [10].

GAP programı kullanarak mertebesi yani eleman sayısı dört olan grupların değişmeli olduğu aşağıdaki gibi görülebilir.

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> Size(G);
4
gap> IsAbelian(G);
true
gap> H:=Group((1,4,2,3));
Group([ (1,4,2,3) ])
gap> Size(H);
4
gap> IsAbelian(H);
true
gap> U:=Group((1,2),(3,4));
Group([ (1,2), (3,4) ])
gap> Size(U);
4
gap> IsAbelian(U);
true
```

Teorem : G ve G' iki grup olsun. $G\times G'$ kümesi, $x,x'\in G$ ve $y,y'\in G'$ için

$$(x,y)(x',y')=(xx',yy')$$

işlemine göre bir gruptur. Bu gruba G ve G' nün direkt çarpımı denir [7].

GAP programında iki grubun direkt çarpımını hesaplamak için `DirectProduct()` komutu kullanılır. Buradaki `()` işaretleri arasına direkt çarpımı hesaplanacak olan gruplar yazılmalıdır [17].

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> H:=Group((1,2)(3,4));
Group([ (1,2)(3,4) ])
gap> D:=DirectProduct(G,H);
Group([ (1,2,3,4), (5,6)(7,8) ])
```

```
gap> IsGroup(D);
true
gap> Order(G);
4
gap> Order(H);
2
gap> Order(D);
8
```

Örnek : Değişmeli iki grubun direkt çarpımını da değişmeli olduğunu gösterelim [7].

```
gap> IsAbelian(G);
true
gap> IsAbelian(H);
true
gap> IsAbelian(D);
true
```

9.3 Permütasyon Grubu

$A \neq \emptyset$ bir küme olmak üzere $S(A) = \{\sigma : A \to A$ birebir örten fonksiyon $\}$ kümesini tanımlayalım. Her $\sigma, \tau \in S(A)$ ve $x \in A$ için

$$(\sigma o \tau)(x) = \sigma(\tau(x))$$

şeklinde tanımlanan fonksiyonların bileşke işlemi $S(A)$ kümesi üzerinde ikili işlem olup $(S(A), o)$ bir gruptur. $S(A)$ nın elemanları permütasyon olduğundan bu gruba A üzerinde bir permütasyon grubu denir. Ayrıca $S(A)$ abelyen olması için gerek ve yeter şart A nın en çok iki tane elemanının olmasıdır. A, n elemanlı bir sonlu küme ise $S(A)$ nın mertebesi $n!$ dir [3].

Tanım : Eğer $A = \{1, 2, ..., n\}$ ise yani A kümesi sonlu ise $S(A)$ grubuna n eleman üzerinde bir simetrik grup denir ve kısaca S_n ile gösterilir.

$k \in [1, n]$ olmak üzere,

$$\sigma : A \to A$$
$$k \mapsto \sigma(k) = i_k$$

olarak tanımlanır ve

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & \dots & n \\ i_1 & i_2 & i_3 & \dots & i_n \end{pmatrix}$$

şeklinde yazılır [3].

GAP programında simetrik gruplar `SymmetricGroup(IsPermGroup,n)` komutu ile oluşturulurlar [17]. Simetrik grup ile ilgili temel özellikler daha detaylı olarak Bölüm 13 de verilecektir. Şimdi GAP programını kullanarak S_4 simetrik grubunu oluşturalım.

```
gap> s4:=SymmetricGroup(IsPermGroup,4);
Sym( [ 1 .. 4 ] )
```

Teorem : Düzgün bir n-genin herhangi bir simetrisi bu n-genin simetri merkezini sabit bırakır [4].

İspat :Düzgün n-genin simetri merkezi, bu n-genin çevrel çemberinin merkezidir. Merkezi 0 ve çevrel çemberinin yarıçapı R olan bu n-geni S ile gösterelim. f, S nin herhangi bir simetrisi ve $x \in S$ olsun. Bu x in 0 merkezine olan uzaklığı r_x ise $f : S \to S$ bir simetri olduğundan $f(x)$ in $f(0)$ a olan uzaklığı da r_x dir. S nin köşe noktalarının merkeze olan uzaklığı R olduğu için *f(S)* ninde $f(0)$ dan R uzaklığında olan noktaları vardır. Fakat $f(S) = A$ dir. O halde $f(0)$ merkezli ve R yarıçaplı çember S nin bir çevrel çemberidir. S nin çevrel çemberi tek olduğundan $f(0) = 0$ dır [4].

Teorem : Düzgün bir n-genin her bir simetrisi n-genin köşegenlerini birbirine dönüştürür [4].

İspat :Düzgün n-geni S ile gösterelim. S nin, çevrel çemberinin yarıçapı R, merkezi 0 ve herhangi bir köşesi de A olup $f(0) = 0$ dır. f uzaklıkları koruduğu için $|f(0) - f(A)| = |0 - f(A)| = |0 - A|$ yazılabilir. O halde $f(A)$ nın 0 a olan uzaklığı R dir. Aynı zamanda $f(A) \in S = f(S)$ olduğundan $f(A)$, S nin bir köşe noktasıdır. Çünkü S nin, merkezden R uzaklığında olan noktaları sadece köşe noktalarıdır [4].

Tanım: Düzgün bir n-genin köşelerini $A_1, A_2, ..., A_n$ ile gösterelim. Bu n-genin herhangi bir simetrisi f ise mesela A_1 köşesinin f altındaki görüntüsü $A_1, A_2, ..., A_n$ köşesinden biridir. Fakat A_2 köşesi için $|f(A_1) - f(A_2)| = |A_1 - A_2|$ olacağından $f(A_2)$ için iki ihtimal vardır. Yani $f(A_2)$, $f(A_1)$ e komşu köşelerden biri olmak zorundadır. $f(A_1)$ ve $f(A_2)$ nin belli olması ile f dönüşümü de belli olur. O halde düzgün n-genin bütün simetrilerinin sayısı $2n$ dir. Bu simetriler kümesi, bileşke işlemi altında bir grup oluşturur. Bu gruba "Dihedral Grup" denir ve D_n ile gösterilir [4].

GAP programında D_n Dihedral Grubunu oluşturmak için kullanılacak komut `DihedralGroup(IsPermGroup,2n)` şeklindedir. Örneğin D_8 dihedral grubu şu şekilde oluşturulur [17].

```
gap> d8:=DihedralGroup(IsPermGroup,16);
Group([ (1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6) ])
```

Yukarıdaki örnekte oluşturulan grup içerisinde verilen (1,2,3,4,5,6,7,8) ve (2,8)(3,7)(4,6) elemanları grubun üreteçleridir.

Bir G grubun üreteçlerinin elde edilebilmesi için GAP programında kullanılan komut `GeneratorsOfGroup(G)` şeklindedir [17]. Örneğin oluşturulan D_8 dihedral grubunun üreteçlerini liste halinde görelim.

```
gap> GeneratorsOfGroup(d8);
[ (1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6) ]
```

D_8 dihedral grubunun eleman sayısını hesaplayalım.

```
gap> Size(d8);
16
gap> Order(d8);
16
```

D_8 dihedral grubunun elemanlarını listeleyelim.

```
gap> Elements(d8);
[(),(2,8)(3,7)(4,6),(1,2)(3,8)(4,7)(5,6),(1,2,3,4,5,6,7,8),
(1,3)(4,8)(5,7),(1,3,5,7)(2,4,6,8),(1,4)(2,3)(5,8)(6,7),
(1,4,7,2,5,8,3,6),(1,5)(2,4)(6,8),(1,5)(2,6)(3,7)(4,8),
(1,6)(2,5)(3,4)(7,8),(1,6,3,8,5,2,7,4),1,7)(2,6)(3,5),
(1,7,5,3)(2,8,6,4),(1,8,7,6,5,4,3,2),(1,8)(2,7)(3,6)(4,5)]
```

Bir cebirsel yapının grup olup olmadığı `IsGroup` komutu ile sorgulanabilir [17].

```
gap> IsGroup(d8);
true
```

Simetrik grupların nasıl oluşturulduğundan daha önce bahsedilmişti. Özel olarak S_3 simetrik grubu D_3 dihedral grubuna eşittir. Şimdi bu iki grubu oluşturup eşitliğini inceleyelim.

```
gap> d3:=DihedralGroup(IsPermGroup,6);
Group([ (1,2,3), (2,3) ])
gap> s3:=SymmetricGroup(IsPermGroup,3);
Sym( [ 1 .. 3 ] )
gap> Size(s3);
6
gap> Size(d3);
6
gap> Elements(s3);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
gap> Elements(d3);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
gap> s3=d3;
true
```

GAP programı kullanılarak permütasyon çarpımı yapılabilir. GAP programı çarpım işlemini soldan sağa doğru yapar ancak matematik kitaplarında bu çarpım sağdan sola doğru yapılır. Yani bir matematik kitabında sağdan sola doğru yapılan permütasyon çarpımı GAP programı yardımıyla hesaplanmak istenirse elemanlar yer değiştirilerek yazılmalıdır.

Örnek : $\alpha = (1\ 4\ 3\ 2)$ ve $\beta = (1\ 3\ 2)$ olmak üzere ;

$$\alpha\beta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix} = (1\ 2\ 4\ 3)$$

olur [14].

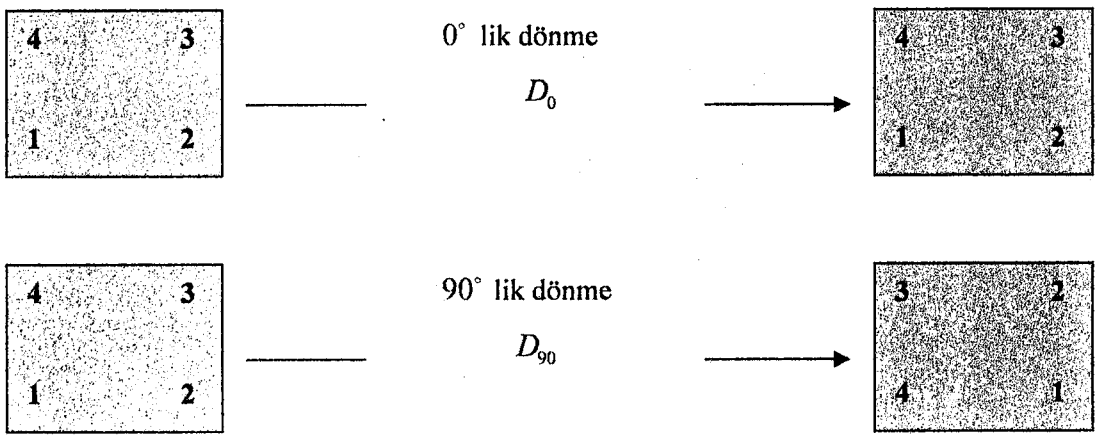
Bu örneği GAP programı kullanarak tekrar inceleyelim.

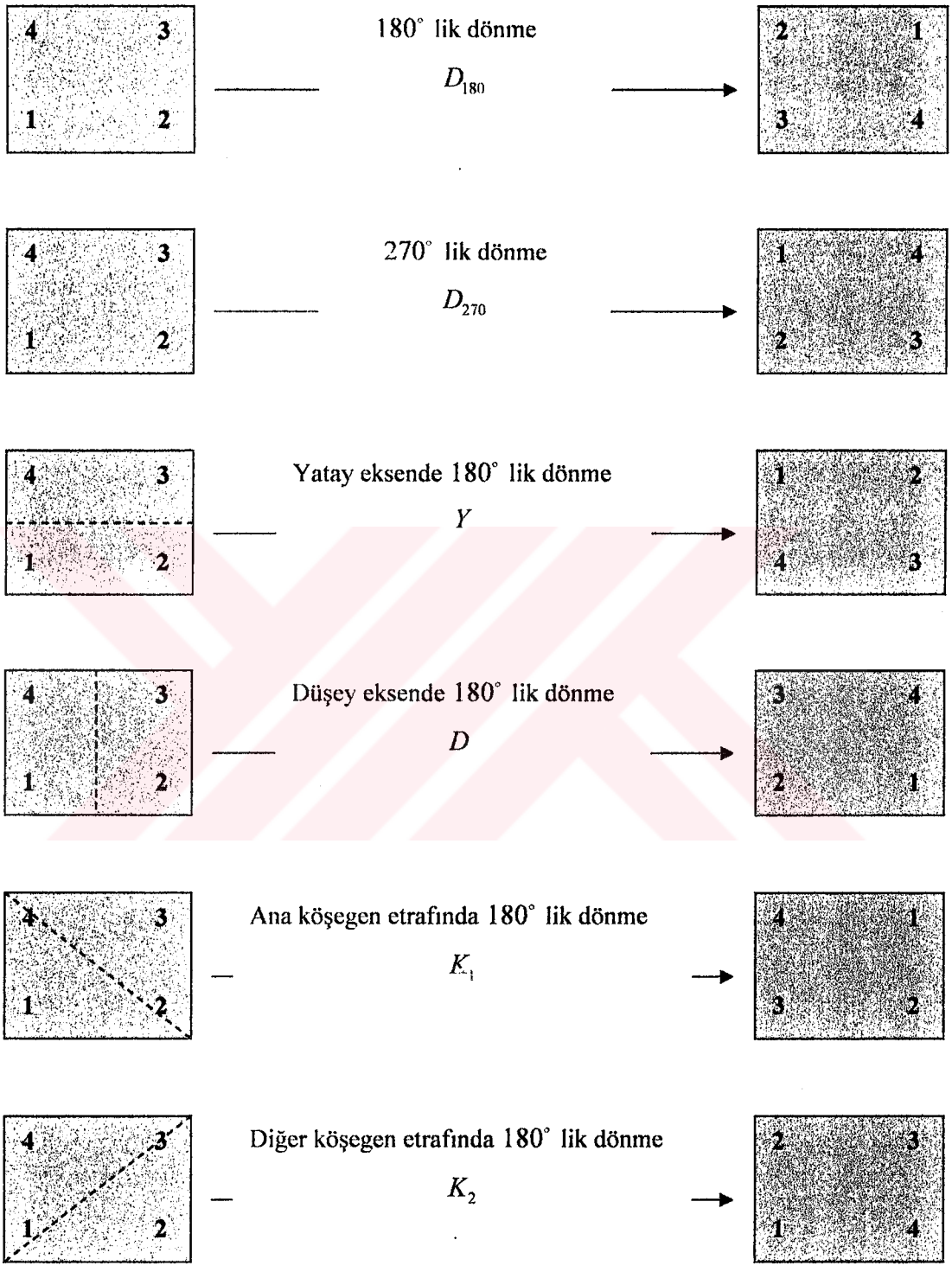
```
gap> (1,3,2)*(1,4,3,2);
(1,2,4,3)
```

D_3 dihedral grubundaki (1 2 3) elemanının tersi bulunmak istenirse bu durumda elemanın tersi ile kendisinin çarpımı birim elemanı vermek zorundadır.

```
gap> (1,2,3)^(-1);
(1,3,2)
gap> (1,2,3)*(1,3,2);
()
gap> (1,3,2)*(1,2,3);
()
```

D_4 dihedral grubu abelyen olmayan ve sekizinci mertebeden bir gruptur. Bu grup bir karenin elemanlarının simetri merkezine, eksenlere ve köşegenlere göre dönmeleri sonucunda elde edilirler.





Şekil 9.1 D_4 dihedral grubu

D_4 dihedral grubu $\{D_0, D_{90}, D_{180}, D_{270}, Y, D, K_1, K_2\}$ elemanlarından oluşur. Bu elemanlar;

$$D_0=\begin{pmatrix}1&2&3&4\\1&2&3&4\end{pmatrix}\ D_{90}=\begin{pmatrix}1&2&3&4\\4&1&2&3\end{pmatrix}\ D_{180}=\begin{pmatrix}1&2&3&4\\3&4&1&2\end{pmatrix}\ D_{270}=\begin{pmatrix}1&2&3&4\\2&3&4&1\end{pmatrix}$$

$$Y=\begin{pmatrix}1&2&3&4\\4&3&2&1\end{pmatrix}\ D=\begin{pmatrix}1&2&3&4\\2&1&4&3\end{pmatrix}\ K_1=\begin{pmatrix}1&2&3&4\\3&2&1&4\end{pmatrix}\ K_2=\begin{pmatrix}1&2&3&4\\1&4&3&2\end{pmatrix}$$

elemanlarından ibarettir [9].

```
gap> d4:=DihedralGroup(IsPermGroup,8);
Group([ (1,2,3,4), (2,4) ])
gap> ed4:=Elements(d4);
[(),(2,4),(1,2)(3,4),(1,2,3,4),(1,3),(1,3)(2,4),(1,4,3,2),
(1,4)(2,3) ]
gap> Size(d4);
8
gap> D0:=ed4[1];
()
gap> D90:=ed4[7];
(1,4,3,2)
gap> D180:=ed4[6];
(1,3)(2,4)
gap> D270:=ed4[4];
(1,2,3,4)
gap> Y:=ed4[8];
(1,4)(2,3)
gap> D:=ed4[3];
(1,2)(3,4)
gap> K1:=ed4[5];
(1,3)
gap> K2:=ed4[2];
(2,4)
gap> GeneratorsOfGroup(d4);
[ (1,2,3,4), (2,4) ]
gap> IsAbelian(d4);
false
```

```
gap> K1*K2=K2*K1;
true
gap> D*Y=Y*D;
true
gap> D270*D90=D90*D270;
true
gap> D180*D90=D180*D270;
false
```

10 ALT GRUPLAR

10.1 Giriş

Bu bölüm alt gruplara ayrılmıştır. Bu bölümde de alt grupların tanım, teorem ve örnekleri GAP programı aracılığı ile incelenmiştir. Her bölümde olduğu gibi bu bölümde de GAP programı notasyonları tanıtılmıştır. Bu GAP programı fonksiyonlarının kullanım formatları da bölüm içerisinde sunulmuştur. Burada en önemli GAP programı fonksiyonları `Subgroup`, `IsSubgroup`, `Centre`, `Centralizer`, `Normalizer`, `Order` fonksiyonlarıdır.

Ayrıca bir grubun alt grubunun bulunmasındaki yöntemlerin yanı sıra grup latisleride bu bölüm içerisinde düşünülmüş olup bu grup latis çizimleri de 30. Dereceden gruplar için bu çalışmanın sonunda sunulmuştur.

10.2 Alt Grup Kavramı

Tanım : G bir grup olsun. Eğer

i) $\varnothing \neq H \subseteq G$,

ii) H , G grubunun işlemi altında bir grup,

özellikleri sağlanıyorsa H ya G grubunun bir alt grubu denir (veya G nin aynı işlem altında boş olmayan bir G altkümesine kısıtlanmışıdır denir) ve $H \leq G$ ile gösterilir [3].

GAP programında `Subgroup()` fonksiyonu alt grup elde etmek için kullanılır. `()` işaretleri arasına alt grubu oluşturulacak grup ile alt grup için üreteçler bir liste halinde yazılmalıdır [17].

```
gap> G:=Group((1,2,3,4),(1,2,3));
Group([ (1,2,3,4), (1,2,3) ])
gap> Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),
(1,2)(3,4),(1,2,3),(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),
(1,3,4,2), (1,3), (1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),
(1,4,3),(1,4),(1,4,2,3),(1,4)(2,3)]
gap> Subgroup(G,[(1,2,3,4)]);
Group([ (1,2,3,4) ])
```

```
gap> H:=Subgroup(G,[(1,2,3,4)]);
Group([ (1,2,3,4) ])
gap> L:=Subgroup(G,[(1,2,5)]);
Error, <gens> must be elements in <M> called from <function>(
<arguments> ) called from read-eval-loop Entering break read-
eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
gap> L:=SubgroupNC(G,[(1,2,5)]);
Group([ (1,2,5) ])
```

Burada dikkat edilecek nokta, alt gruplar için yazılacak üreteçlerin grubun bir elemanı olması gerektiğidir. Aksi durumda GAP programı yukarıda olduğu gibi hata verecektir. Bu gibi hataların oluşmasını engellemek için GAP programında ikinci bir alt grup fonksiyonu daha vardır ki bu fonksiyon `SubgroupNC()` fonksiyonudur. Bu fonksiyonun kullanım şekli de tıpkı `Subgroup()` fonksiyonu gibidir. Bu iki fonksiyon arasındaki tek fark `SubgroupNC()` fonksiyonunda kullanılacak olan üreteçlerin grubun elemanı olup olmadığının incelenmemesidir. Kullanılan üreteçler aynı olduğu sürece `Subgroup()` ve `SubgroupNC()` fonksiyonları aynı işlevi gerçekleştirecek yani aynı grubu oluşturacaklardır.

```
gap> g:=Group([(1,2,3,4),(1,2)]);
Group([ (1,2,3,4), (1,2) ])
gap> Elements(g);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> v:=Subgroup(g,[(1,4)(2,3)]);
Group([ (1,4)(2,3) ])
gap> IsGroup(v);
true
gap> w:=SubgroupNC(g,[(1,4)(2,3)]);
Group([ (1,4)(2,3) ])
gap> IsGroup(w);
```

```
true
gap> w=v;
true
gap> w:=SubgroupNC(g,[(1,5)(2,3)]);
Group([ (1,5)(2,3) ])
gap> w=v;
false
gap> IsGroup(w);
true
```

Teorem : G bir grup ve H da G nin boş olmayan bir alt kümesi olsun. Aşağıdaki ifadeler birbirine denktir.

i) H, G nin bir alt grubu,

ii) H aşağıdaki özellikleri sağlar;

a) $e \in H$

b) $x, y \in H \Rightarrow xy \in H$

c) $x \in H \Rightarrow x^{-1} \in H$

iii) $x, y \in H \Rightarrow xy^{-1} \in H$ [3]

Teorem: U ve U', G grubunun iki alt grubu olsun. $U \cap U'$ kümeside G nin bir alt grubudur [5].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> g:=Group([(1,2,3,4),(1,2)]);
Group([ (1,2,3,4), (1,2) ])
gap> Elements(g);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> v:=Subgroup(g,[(1,4,3,2)]);
Group([ (1,4,3,2) ])
gap> z:=Subgroup(g,[(1,4)(2,3)]);
Group([ (1,4)(2,3) ])
```

```
gap> Intersection(v,z);
Group(())
```

IsSubgroup() fonksiyonu () işaretleri arasına yazılacak olan iki adet gruptan ikincisinin birincisinin alt grubu olup olmadığını inceler. IsSubgroup() fonksiyonu IsGroup() ve IsSubset() fonksiyonlarının birleşmiş halidir. Bunun anlamı. IsSubgroup() fonksiyonundaki () işaretleri arasına yazılacak grupların ilk önce birbirlerinin kümesi olup olmadığı sonra da bu kümenin bir grup yapısı oluşturup oluşturmadığı incelenir. Bu iki sorgulamanın sonucu da doğru çıkarsa IsSubgroup() fonksiyonunun sonucu da doğru çıkacaktır [17].

```
gap> G:=Group((1,2,3,4),(1,2));
Group([ (1,2,3,4), (1,2) ])
gap> H:=Subgroup(G,[(1,4,3,2)]);
Group([ (1,4,3,2) ])
gap> IsSubset(G,H);
true
gap> IsGroup(H);
true
gap> IsSubgroup(G,H);
true
gap> U:=Subgroup(G,[(1,4)(2,3)]);
Group([ (1,4)(2,3) ])
gap> IsSubset(G,U);
true
gap> IsGroup(U);
true
gap> IsSubgroup(G,U);
true
gap> IsSubset(H,U);
false
gap> IsGroup(U);
true
gap> IsSubgroup(H,U);
false
```

Tanım : G bir grup ve A ve B, G grubunun iki alt kümesi olsunlar.

$$AB = \{ab : a \in A, b \in B\}$$

ye A ile B kümelerinin çarpımı denir. Özel olarak, $A = \{a\}$ ise $\{a\}B = aB$ ile gösterilir [7].

Teorem : $H, K < G$ olsun.

$$HK < G \Leftrightarrow HK = KH$$

özelliği sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=Subgroup(d4,[ed4[1]]);
Group(())
gap> K:=Subgroup(d4,[ed4[8]]);
Group([ (1,4)(2,3) ])
gap> HK:=DirectProduct(H,K);
Group([ (), (1,4)(2,3) ])
gap> IsSubgroup(d4,HK);
true
gap> DirectProduct(H,K)=DirectProduct(K,H);
true
```

Tanım: G bir grup olsun.

$$M(G) = Mer_G(G) = \{x \in G : \text{her g} \in G \text{ için } xg = gx\}$$

alt grubuna, G nin merkezi denir [3].

GAP programında `Centre()` fonksiyonu bir grubun merkezini oluşturmak için kullanılır. Buradaki `()` işaretleri arasına merkezi oluşturulacak olan grup yazılmalıdır. `Centre()` fonksiyonu yerine `Center()` fonksiyonu da kullanılabilir. Bu iki fonksiyondan birisi ile oluşturulan yeni grup mutlaka abelyen olmak zorundadır. Çünkü bu grup içerisindeki değişmeli olan elemanlar seçilip yeni bir grup oluşturulur [17].

```
gap> G:=Group((1,2,3,4),(1,3));
Group([ (1,2,3,4), (1,3) ])
gap> M:=Centre(G);
Group([ (1,3)(2,4) ])
gap> M:=Center(G);
```

```
Group([ (1,3)(2,4) ])
gap> IsAbelian(M);
true
```

Grupta birim eleman dışında hiçbir eleman değişme özelliğini sağlamıyorsa `Centre()` (veya `Center()`) fonksiyonu oluşturulabilecek olan en küçük grubu yani sadece birim elemandan oluşan grubu oluşturacaktır.

```
gap> G:=Group((1,2,3,4),(1,2,3));
Group([ (1,2,3,4), (1,2,3) ])
gap> M:=Center(G);
Group(())
gap> M:=Centre(G);
Group(())
IsAbelian(M);
true
```

Tüm elemanları değişmeli olan, yani abelyen bir grup alındığında bu grubun merkezi grubun kendisi olacaktır.

```
gap> G:=Group((1,2,3),(4,5,6));
Group([ (1,2,3), (4,5,6) ])
gap> IsAbelian(G);
true
gap> M:=Center(G);
Group([ (4,5,6), (1,2,3) ])
gap> IsAbelian(M);
true
gap> M=G;
true
```

Bir grubun merkezinin, aynı zamanda o grubun alt grubu olduğunu GAP programı kullanarak görelim.

```
gap> G:=Group((1,2,3,4),(1,3));
Group([ (1,2,3,4), (1,3) ])
```

```
gap> M:=Center(G);
Group([ (1,3)(2,4) ])
gap> IsSubgroup(G,M);
true
```

Tanım : G bir grup ve $a \in G$ olsun.

$$M(a) = \{g \in G : ag = ga\}$$

kümesine a nın merkezleştiricisi denir [7].

GAP programında `Centralizer()` komutu grup içerisindeki bir elemanın merkezleştiricisini bulmak için kullanılır. `()` işaretleri arasına bir grup ve bu grup içerisinde merkezleştiricisi bulunacak olan eleman yazılmalıdır [17].

```
gap> G:=Group((1,3,2),(2,4));
Group([ (1,3,2), (2,4) ])
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> a:=eG[8];
(1,2)(3,4)
gap> MG:=Centralizer(G,a);
Group([ (3,4), (1,2), (1,3)(2,4) ])
```

GAP programı kullanarak $M(a)$ kümesinin G grubunun bir alt grubu olduğunu gösterelim.

```
gap> IsSubgroup(G,MG);
true
gap> b:=eG[10];
(1,2,3,4)
gap> MU:=Centralizer(G,b);
Group([ (1,2,3,4) ])
gap> IsSubgroup(G,MU);
true
```

Tanım : $H \leq G$ olsun.

$$N(H) = \{g \in G : gh = hg \ , \forall h \in H\}$$

kümesine H ın G içerisindeki normalleştiricisi denir [15].

GAP programında `Normalizer()` komutu bir alt grubun diğer bir grup içerisindeki bir elemanın merkezleştiricisini bulmak için kullanılır. `()` işaretleri arasına bir grup ve bu grup içerisinde merkezleştiricisi bulunacak olan eleman yazılmalıdır. `Normalizer()` komutu yukarıdaki tanımda verilen $H \leq G$ şartının sağlanıp sağlanmadığını test etmez. $H \leq G$ koşulu sağlanmıyorsa `Normalizer()` komutu yalnızca birim elemandan oluşan grubu verecektir [17].

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> eG:=Elements(G);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> H:=Subgroup(G,[(1,3)(2,4)]);
Group([ (1,3)(2,4) ])
gap> IsSubgroup(G,H);
true
gap> NH:=Normalizer(G,H);
Group([ (1,2,3,4) ])
gap> U:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
gap> IsSubgroup(G,U);
false
gap> NH:=Normalizer(G,U);
Group(())
```

GAP programı kullanarak $N(H)$ kümesinin G grubunun bir alt grubu olduğunu gösterelim.

```
gap> IsSubgroup(G,NH);
true
gap> K:=Subgroup(G,[(1,4,3,2)]);
Group([ (1,4,3,2) ])
```

```
gap> IsSubgroup(G,K);
true
gap> NK:=Normalizer(G,K);
Group([ (1,4,3,2), (1,3)(2,4) ])
gap> IsSubgroup(G,NK);
true
```

Tanım : G bir grup ve $g \in G$ olsun. g nin ürettiği grubun mertebesine g elemanının mertebesi denir ve $|g|$ ile gösterilir. Başka bir ifadeyle $|g|$, $g^n = e$ koşulunu sağlayan $n > 0$ tam sayıları arasında en küçük olanıdır [7].

`Order()` ve `Size()` komutlarının GAP programında bir grubun mertebesini bulmak için kullanıldığından daha önce bahsedilmişti. `Order()` komutu bir elemanın mertebesini bulmak için kullanılır. `Size()` komutu ise sadece bir grubun mertebesini bulmak için kullanılabilir. `Size()` komutu kullanılarak bir elemanın mertebesi bulunmak istenirse GAP programı hata verecektir [17].

```
gap> G:=Group((1,3,2),(1,2));
Group([ (1,3,2), (1,2) ])
gap> eG:=Elements(G);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
gap> Order(eG[2]);
2
gap> eG[2]^2=Identity(G);
true
gap> Order(eG[4]);
3
gap> eG[4]^3=Identity(G);
true
gap> Size(eG[4]);
Error, no method found! For debugging hints type ?Recovery from
NoMethodFound
Error, no 1st choice method found for `Size' on 1 arguments
called from
<function>( <arguments> ) called from read-eval-loop
```

```
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

Örnek :

$$\tau = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} \in S_3 \text{ ve } \beta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} \in S_4$$

elemanlarının mertebelerini bulalım.

$\tau^2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$ olup τ ikinci mertebedendir.

$$\beta^2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

$$\beta^3 = \beta^2\beta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}$$

$\beta^4 = \beta^3\beta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$ olup β dördüncü mertebedendir.

β ve τ elemanlarının mertebelerini GAP programı kullanarak inceleyelim.

```
gap> to:=(1,2);
(1,2)
gap> Order(to);
2
gap> to^2;
()
gap> beta:=(1,4,3,2);
(1,4,3,2)
gap> Order(beta);
4
gap> beta^4;
()
```

Teorem : G bir grup, $a \in G$ ve $|a| = n$ olsun.

$$a^m = e \Leftrightarrow n \mid m$$

özelliği sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim. n sayısının m sayısını bölmesi demek m sayısı n ile tam olarak bölünmesi demektir.

```
gap> G:=Group((1,4),(1,2,3));
Group([ (1,4), (1,2,3) ])
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> a:=eG[5];
(2,4,3)
gap> Order(a);
3
gap> a^12;
()
gap> 12/3;
4
```

Örnek : G bir grup ve $a \in G$ olmak üzere

$$|a| = |a^{-1}|$$

eşitliğinin sağlandığını gösterelim [7].

```
gap> a:=eG[8];
(1,2)(3,4)
gap> Order(a)=Order(a^-1);
true
```

Örnek : G bir grup ve $a, b \in G$ olmak üzere

$$|a| = |bab^{-1}|$$

eşitliğinin sağlandığını gösterelim [7].

GAP programında işlemlerin soldan sağa doğru yapıldığından bab^{-1} işlemini GAP programında $b^{-1}ab$ şeklinde gerçekleştirmemiz gerekir. .

```
gap> a:=eG[8];
(1,2)(3,4)
gap> b:=eG[12];
(1,2,4)
gap> Order(a)=Order(b^-1*a*b);
true
gap> a:=eG[18];
(1,3,2,4)
gap> b:=eG[15];
(1,3)
gap> Order(a)=Order(b^-1*a*b);
true
```

Örnek : G bir grup ve $a \in G$ ise $|a| = n$ ise $a^r = a^s$ olması için gerek ve yeter koşul

$$r \equiv s \pmod{n}$$

olması gerektiğini gösterelim [7].

```
gap> a:=eG[10];
(1,2,3,4)
gap> n:=Order(a);
4
gap> s:=35;
35
gap> r:=s mod n;
3
gap> a^r=a^s;
true
```

Örnek : G değişmeli bir grup ve $a,b \in G$ olsun. $|a| = m$, $|b| = n$ ve $(m,n) = 1$ olmak üzere

$$|ab| = mn$$

eşitliğinin sağlandığını gösterelim [7].

```
gap> G:=Group((1,2,3,4,5,6));
Group([ (1,2,3,4,5,6) ])
gap> IsAbelian(G);
true
gap> eG:=Elements(G);
[(),(1,2,3,4,5,6),(1,3,5)(2,4,6),(1,4)(2,5)(3,6),(1,5,3)(2,6,4),
(1,6,5,4,3,2)]
gap> a:=eG[3];
(1,3,5)(2,4,6)
gap> b:=eG[4];
(1,4)(2,5)(3,6)
gap> m:=Order(a);
3
gap> n:=Order(b);
2
gap> Gcd(m,n);
1
gap> Order(a*b)=m*n;
true
```

Örnek : G bir grup, $a \in G$ ve $|a| = n$ olmak üzere

$$|a^m| = \frac{n}{(n,m)}$$

eşitliğinin sağlandığını gösterelim [7].

```
gap> a:=eG[4];
(1,4)(2,5)(3,6)
gap> n:=Order(a);
2
gap> m:=10;
10
gap> Order(a^m)=n/Gcd(m,n);
true
```

Örnek : G bir grup ve birim elemandan farklı $x \in G$ elemanları için

$$|a| = 2$$

eşitliği sağlanıyorsa G grubunun değişmeli olduğunu gösterelim.

```
gap> G:=Group((1,2),(1,2));
Group([ (1,2), (1,2) ])
gap> eG:=Elements(G);
[ (), (1,2) ]
gap> a:=eG[2];
(1,2)
gap> a=Identity(G);
false
gap> Order(a);
2
gap> IsAbelian(G);
true
```

Örnek : $(\mathbb{Z}_{18}, +)$ grubunun tüm alt gruplarını bulalım ve latisini çizelim.

$n = 18$ olmak üzere $d \mid n = d \mid 18 \Rightarrow d = 1,2,3,6,9,18$ olup alt gruplar $< \frac{n}{d}.a >$ formundadır.

$$H_1 = < \frac{18}{1}.1 > = \{0\}$$

$$H_2 = < \frac{18}{2}.1 > = \{0,9\}$$

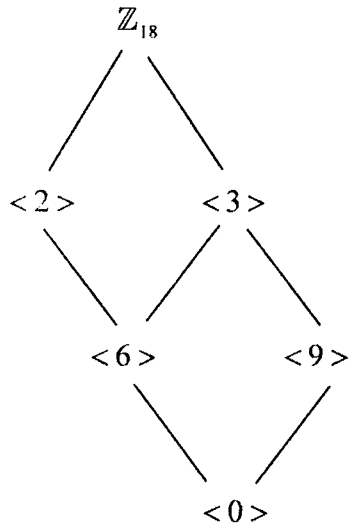
$$H_3 = < \frac{18}{3}.1 > = \{0,6,12\}$$

$$H_4 = < \frac{18}{6}.1 > = \{0,3,6,9,12,15\}$$

$$H_5 = < \frac{18}{9}.1 > = \{0,2,4,6,8,,10,12,14,16\}$$

$$H_6 = < \frac{18}{18}.1 > = \mathbb{Z}_{18}$$

olur.


$\mathbb{Z}_{18}$
<2>
<3>
<6>
<9>
<0>

11 NORMAL ALT GRUPLAR

11.1 Giriş

Bu bölümde bir grubun normal alt gruplarının bulunması, yan cümleler, bir grubun merkezi, normalleştirici. Lagrange teoremi, index, Euler teoremi, Fermat teoremi, bölüm grubu, komütatör alt grubu kavramlarının bilgisayara uygulama fonksiyonları olan `NormalSubgroups()`, `RightCosets()`, `Center()`, `Normalizer()`, `Index()`, `FactorGroup()`, `CommutatorSubgroup()`, `MaximalSubgroup()` gibi fonksiyonlarının kullanılmasına örnekler verilmiş olup bu fonksiyonların GAP programında kullanma yöntemleride bu bölümde sunulmuştur.

11.2 Normal Alt Grup Kavramı

Tanım : G bir grup ve $H \leq G$ olsun. G de $\equiv$ bağıntısını,

$$x \equiv y \pmod{H} \Leftrightarrow xy^{-1} \in H$$

şeklinde tanımlayalım. Yukarıda tanımlanan $\equiv$ bir denklik bağıntısıdır [7].

Tanım : G bir grup ve $H \leq G$ olsun. $x \in G$ olmak üzere

$$Hx = \{hx : h \in H\}$$

kümesine H nın G deki x tarafından belirlenen sağ denklik sınıfı denir [7].

Tanım : G bir grup ve $H \leq G$ olmak üzere

$$x \equiv y \pmod{H} \Leftrightarrow x^{-1}y \in H$$

ile tanımlı $\equiv$ bağıntısı G de bir denklik bağıntısıdır. Bu denklik bağıntısına göre, $x \in G$ elemanının sınıfı $xH = \{xh : h \in H\}$ alt kümesidir. xH ya H alt grubuna göre x in sol denklik sınıfı denir [7].

Uyarı :

i) H alt grubunun kendisi de bir denklik sınıfıdır. Çünkü $H = He$ dir.

ii) H alt grup ve $e \in H$ olduğundan $x = ex \in Hx$ dir. Böylece $x \in G$ elemanı her zaman Hx (veya xH sol) denklik sınıfının elemanıdır.

iii) Her zaman iki tane denklik sınıf vardır. Birincisi $H = G$ ise H nın kendisidir. İkincisi $H = \{e\}$ ise her $x \in G$ için $Hx = \{x\}$ dir.

GAP programında denklik sınıfları `ConjugacyClasses()` fonksiyonu kullanılarak bulunur. Buradaki `()` işaretleri arasına denklik sınıfları bulunacak olan grup yazılmalıdır. Bu fonksiyon bir G grubunun denklik sınıflarını bir küme halinde verecektir. Yukarıdaki uyarıdan da anlaşılacağı gibi bu küme en az iki elemanlıdır. `NrConjugacyClasses()` komutu ise bir grubunun denklik sınıfları sayısını verecektir.

```
gap> G:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
gap> ConjugacyClasses(G);
[()^G,(1,2,3,4,5)^G,(1,3,5,2,4)^G,(1,4,2,5,3)^G,(1,5,4,3,2)^G ]
gap> NrConjugacyClasses(G);
5
```

Teorem : $N \leq G$ olmak üzere,

i) $\forall a \in G$, $\forall x \in N$ için $axa^{-1} \in N$ dir

ii) $\forall a \in G$ için $aNa^{-1} \subset N$ dir.

iii) $\forall a \in G$ için $aNa^{-1} = N$ dir.

iv) $\forall a \in G$ için $aN = Na$ dır.

ifadeleri birbirine denktir [7].

Tanım : Yukarıdaki teoremin denk koşullarından birini sağlayan G nin bir N grubuna **normal alt grup** denir ve $N \lhd G$ ile gösterilir. Burada $Na = \{nx : n \in N\}$ alt kümesine sağ yan küme denir [7].

GAP programında `NormalSubgroups()` komutu bir grubun tüm normal alt gruplarını bulmak için kullanılır. `()` işaretleri arasına normal alt grupları bulunacak olan grup yazılmalıdır [17].

```
G:=Group((1,2,3),(2,3),(4,5,6,7,8));
Group([ (1,2,3), (2,3), (4,5,6,7,8) ])
gap> NormalSubgroups(G);
[Group([(1,2,3),(2,3),(4,5,6,7,8)]),Group([(4,5,6,7,8),(1,2,3)
]),Group([(2,3),(1,2,3)]),Group([(4,5,6,7,8)]),Group([(1,2,3)]),
Group(()) ]
```

GAP programında `RightCoset()` komutu grubun bir elemanına göre sağ yan kümeyi bulmak için kullanılır. `()` işaretleri arasına sırasıyla bir alt grup ve bu alt grubu kapsayan gruptan bir eleman yazılmalıdır [17].

```
gap> eG:=Elements(G);
[(),(4,5,6,7,8),(4,6,8,5,7),(4,7,5,8,6),(4,8,7,6,5),(2,3),(2,3)
(4,5,6,7,8),(2,3)(4,6,8,5,7),(2,3)(4,7,5,8,6),(2,3)(4,8,7,6,5),
(1,2),(1,2)(4,5,6,7,8),(1,2)(4,6,8,5,7),(1,2)(4,7,5,8,6),(1,2)
(4,8,7,6,5),(1,2,3),(1,2,3)(4,5,6,7,8),(1,2,3)(4,6,8,5,7),
(1,2,3)(4,7,5,8,6),(1,2,3)(4,8,7,6,5),(1,3,2),(1,3,2)(4,5,6,7,8)
,(1,3,2)(4,6,8,5,7),(1,3,2)(4,7,5,8,6),(1,3,2)(4,8,7,6,5),(1,3),
(1,3)(4,5,6,7,8),(1,3)(4,6,8,5,7),(1,3)(4,7,5,8,6),(1,3)
(4,8,7,6,5) ]
gap> U:=Subgroup(G,[eG[5]]);
Group([ (4,8,7,6,5) ])
gap> RightCoset(U,eG[5]);
RightCoset(Group( [ (4,8,7,6,5) ] ),(4,8,7,6,5))
```

`RightCosets()` komutu grubun bir alt grubuna göre tüm sağ yan kümelerini bulmak için kullanılır. `()` işaretleri arasına sırasıyla bir grup ve bu grubun bir alt grubu yazılmalıdır. `RightCosetsNC()` versiyonu yazılan ikinci grubun birincinin alt grubu olup olmadığını test etmez [17].

```
gap> RightCosets(G,U);
[ RightCoset(Group( [ (4,8,7,6,5) ] ),()),
  RightCoset(Group( [ (4,8,7,6,5) ] ),(2,3)),
  RightCoset(Group( [ (4,8,7,6,5) ] ),(1,3,2)),
  RightCoset(Group( [ (4,8,7,6,5) ] ),(1,3)),
  RightCoset(Group( [ (4,8,7,6,5) ] ),(1,2,3)),
  RightCoset(Group( [ (4,8,7,6,5) ] ),(1,2)) ]
```

Örnek : G bir grup olmak üzere

$$M(G) = Mer_G(G) = \{x \in G : \text{her g} \in G \text{ için } xg = gx\}$$

alt grubuna, G nin merkezi denir. $M(G)$ ve $M(G)$ nin her alt grubunun G de bir normal alt grup olduğunu gösterelim.

```
gap> G:=Group((1,3,2),(1,2,3));
Group([ (1,3,2), (1,2,3) ])
gap> M:=Center(G);
Group([ (1,2,3) ])
gap> eM:=Elements(M);
[ (), (1,2,3), (1,3,2) ]
gap> MT:=Subgroup(M,[(1,3,2)]);
Group([ (1,3,2) ])
gap> IsSubgroup(M,MT);
true
gap> NormalSubgroups(G);
[ Group([ (1,3,2), (1,2,3) ]), Group(()) ]
gap> IsNormal(G,M);
true
gap> IsNormal(G,MT);
true
```

Tanım : $H \leq G$ olmak üzere

$$N(H) = \{g \in G : gh = hg \ , \forall h \in H\}$$

kümesine H ın G içerisindeki normalleştiricisi denir. $H \lhd N(H)$ ve $N(H)$ bu özellikte G nin en büyük alt grubudur.

```
gap> G:=Group((1,3,2),(1,4,2,3));
Group([ (1,3,2), (1,4,2,3) ])
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> eG[14];
(1,3,4,2)
gap> H:=Subgroup(G,[eG[14]]);
Group([ (1,3,4,2) ])
gap> NH:=Normalizer(G,H);
Group([ (1,3,4,2), (1,4)(2,3), (2,3) ])
```

```
gap> NormalSubgroups(NH);
[Group([(1,3,4,2),(1,4)(2,3),(2,3)]),Group([(2,3),(1,4)(2,3)]),
Group([(1,3)(2,4),(1,4)(2,3)]),Group([(1,3,4,2),(1,4)(2,3)]),
Group([(1,4)(2,3)]),Group(()) ]
gap> IsNormal(NH,H);
true
```

Tanım : Bir grubun kendisinden ve birim elemandan farklı normal alt gruplarına has normal alt gruplar denir [7].

Teorem (Lagrange Teoremi) : G sonlu bir alt grup ve $H \leq G$ olsun. Bu durumda H nın mertebesi G nin mertebesini böler [7].

GAP programı kullanarak bu teoremi inceleyelim. Bir G grubu ve bu G grubunun alt grubunun alt grubu olacak şekilde bir H grubu alalım. H grubunun mertebesinin G grubunun mertebesini böldüğünü, başka bir deyişle, G grubunun mertebesinin H grubunun mertebesine bölümünün bir tam sayı olduğunu görelim.

```
gap> G:=Group((1,2,3,4)(5,6));
Group([ (1,2,3,4)(5,6) ])
gap> Elements(G);
[ (), (1,2,3,4)(5,6), (1,3)(2,4), (1,4,3,2)(5,6) ]
gap> H:=Group((1,3)(2,4));
Group([ (1,3)(2,4) ])
gap> IsSubgroup(G,H);
true
gap> SG:=Size(G);
4
gap> SH:=Size(H);
2
gap> SG/SH;
2
```

Tanım : $H \leq G$ olsun. H nın G deki sağ (sol) denklik sınıflarının sayısına H nın G deki indeksi denir ve $|G:H|$ ile gösterilir [3].

H nın G deki indeksi

$$|G:H| = \frac{|G|}{|H|}$$

formülü ile hesaplanır.

GAP programında `Index()` ve `IndexNC()` komutları indeks hesaplamalarında kullanılırlar. Burada ki `()` işaretleri arasına indeksi bulunmak istenilen grup ve alt grup yazılmalıdır. `IndexNC()` komutu yazılan ikinci grubun ilkinin alt grubu olup olmadığını incelemez [17].

```
gap> G:=Group((1,2,3,4)(5,6));
Group([ (1,2,3,4)(5,6) ])
gap> Elements(G);
[ (), (1,2,3,4)(5,6), (1,3)(2,4), (1,4,3,2)(5,6) ]
gap> H:=Group((1,4,3,2)(5,6));
Group([ (1,4,3,2)(5,6) ])
gap> U:=Group((1,2));
Group([ (1,2) ])
gap> IsSubgroup(G,H);
true
gap> IsSubgroup(G,U);
false
brk> gap> Index(G,H);
1
gap> IndexNC(G,U);
2
```

Teorem : G sonlu grup olsun. Bu durumda

i) $|G| = |H| \cdot |G:H|$;

ii) $H, K \leq G$ ve $K \subseteq H \subseteq G$ ise

$$|G:K| = |G:H| \cdot |H:K|$$

eşitliği sağlanır [3].

Bu teoremi GAP programı kullanarak incelemek için verilen koşullara uyacak şekilde üç tane grup oluşturalım.

```
gap> G:=Group((1,2,3,4)(5,6));
Group([ (1,2,3,4)(5,6) ])
gap> H:=Group((1,4,3,2)(5,6));
Group([ (1,4,3,2)(5,6) ])
gap> Elements(H);
[ (), (1,2,3,4)(5,6), (1,3)(2,4), (1,4,3,2)(5,6) ]
gap> IsSubgroup(G,H);
true
gap> K:=Group((1,3)(2,4));
Group([ (1,3)(2,4) ])
gap> IsSubgroup(H,K);
true
gap> IsSubgroup(G,K);
true
gap> Size(G)=Size(H)*Index(G,H); Size(G); Size(H); Index(G,H);
true
4
4
1
gap> Index(G,K)=Index(G,H)*Index(H,K);
true
```

Teorem : G , n mertebeden sonlu grup ise her $x \in G$ için

$$x^n = e$$

eşitliği sağlanır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> G:=Group((1,2,3)(4,5)(6,7,8));
Group([ (1,2,3)(4,5)(6,7,8) ])
gap> Size(G);
6
gap> Elements(G);
```

```
[(),(4,5),(1,2,3)(6,7,8),(1,2,3)(4,5)(6,7,8),(1,3,2)(6,8,7),
(1,3,2)(4,5)(6,8,7)]
gap> (4,5)^6;
()
gap> (1,3,2)(6,8,7)^6;
()
```

Teorem (Euler Teoremi) : $a \in G$ olsun. $(a,n)=1$ ise

$$a^{\phi(n)} \equiv 1 \bmod n$$

denkliği sağlanır [3].

Euler teoremini GAP programı kullanarak inceleyelim.

```
gap> a:=7;
7
gap> m:=17;
17
gap> Gcd(a,m);
1
gap> a^Phi(m) mod m = 1;
true
```

Teorem (Fermat Teoremi) : $a \in G$ olsun. p asal sayı ve p , a yı bölmüyorsa

$$a^{p-1} \equiv 1 \bmod p$$

denkliği sağlanır [3].

Fermat Teoremini GAP programı kullanarak inceleyelim. Bunun için bir a sayısı ve bu a sayısını bölmeyen bir p asal sayısı alalım. Başka bir ifadeyle a sayısını p ile böldüğümüzde sonuç tam sayı olmasın.

```
gap> a:=100;
100
gap> p:=13;
13
gap> IsPrime(p);
true
```

```
gap> a/p;
100/13
gap> a^(p-1) mod p = 1;
true
```

Tanım : $N \lhd G$ olsun. G nin N ye göre denklik sınıfları kümesi G/N ile gösterilir. $N \lhd G$ ise G/N de çarpma işlemi $\forall aN, bN \in G/N$ için,

$$(aN)(bN) = (ab)N$$

şeklinde tanımlanır [7].

Teorem : $(aN)(bN) = (ab)N$ çarpımı iyi tanımlıdır [7].

Teorem : $N \lhd G$ ise G/N bir gruptur [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> G:=Group((1,2,3)(4,5)(6,7,8));
Group([ (1,2,3)(4,5)(6,7,8) ])
gap> NormalSubgroups(G);
[Group([(1,2,3)(4,5)(6,7,8)]),Group([(4,5)]),
Group([ (1,3,2)(6,8,7)]),Group(())]
gap> NG:=NormalSubgroups(G);
[Group([(1,2,3)(4,5)(6,7,8)]),Group([(4,5)]),
Group([ (1,3,2)(6,8,7) ]),Group(())]
gap> IsNormal(G,NG[2]);
true
gap> IsGroup(NG[2]);
true
```

Tanım : $N \lhd G$ ise G/N grubuna, G nin N ye göre **bölüm grubu** denir [7].

`FactorGroup()` ve `FactorGroupNC()` komutları GAP programında bölüm grubunu oluşturmak için kullanılır. Buradaki `()` işaretleri arasına grup ve bu grubun bir normal alt grubu yazılmalıdır. `FactorGroupNC()` komutu yazılan ikinci grubun ilkinin normal alt grubu olup olmadığını incelemez. Ancak `FactorGroup()` komutunda ikinci grup birincisinin alt grubu olması gerekmektedir aksi halde GAP programı hata verecektir [17].

```
gap> G:=Group((1,2,3,4)(5,6));
Group([ (1,2,3,4)(5,6) ])
gap> N:=NormalSubgroups(G);
[ Group([ (1,2,3,4)(5,6) ]), Group([ (1,3)(2,4) ]), Group(()) ]
gap> IsNormal(G,N[2]);
true
gap> F:=FactorGroup(G,N[2]);
Group([ f1 ])
gap> IsGroup(F);
true
gap> G:=Group((1,2,3,4),(1,3));
Group([ (1,2,3,4), (1,3) ])
gap> Elements(G);
[(),(2,4),(1,2)(3,4),(1,2,3,4),(1,3),(1,3)(2,4),(1,4,3,2),
(1,4)(2,3) ]
gap> H:=Group((2,4));
Group([ (2,4) ])
gap> IsNormal(G,H);
false
gap> IsSubgroup(G,H);
true
gap> T:=FactorGroupNC(G,H);
Group([ f1, f2 ])
gap> IsGroup(T);
true
```

Bir G grubunun tüm bölüm gruplarını bulmak için bir fonksiyon oluşturalım. Fonksiyon da ilk önce G grubun normal alt gruplarını bulalım. Daha sonra normal alt gruplar listesinin uzunluğunu bulup bu uzunluk sayısınca bir döngü oluşturalım. Bu döngü ile birer birer bölüm gruplarını bulup daha önce oluşturulan boş listeye ekleyerek tüm bölüm grupları listesi elde edilir. Fonksiyonu bir metin editörü kullanarak `fgroup.gi` adı altında oluşturalım.

```
Fgroup := function(G)
local N,F,n,i,glist;
N:=NormalSubgroups(G);
```

```
n:= Length(N);
glist:=[];
for i in [1..n] do
F:=FactorGroup(G,N[i]);
Add(glist,F);
od;
Print("\n",G,  "  Grubu,  ",i,"   tane  Bolum  Grubuna  sahiptir.
Bunlar :  \n" );
return glist;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `fgroup.gi` dosyasını GAP programı aracılığı ile çalıştıralım ve içerisinde yer alan `Fgroup` isimli fonksiyonu kullanarak G grubunun tüm bölüm gruplarını elde edelim.

```
gap> Read("fgroup.gi");
gap> G:=Group((1,2,3),(1,3));
Group([ (1,2,3), (1,3) ])
gap> Fgroup(G);

Group( [ (1,2,3), (1,3) ] )
Grubu, 3  tane Bolum Grubuna sahiptir. Bunlar :
[ Group([  ]), Group([ f1 ]), Group([ (1,2,3), (1,3) ]) ]
```

Teorem : G sonlu bir grup ve $N \lhd G$ ise G/N de sonlu bir grup olmak üzere

$$|G/N| = \frac{|G|}{|N|}$$

eşitliği sağlanır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> G:=Group((1,2,3,4)(5,6));
Group([ (1,2,3,4)(5,6) ])
gap> NG:=NormalSubgroups(G);
[ Group([ (1,2,3,4)(5,6) ]), Group([ (1,3)(2,4) ]), Group(()) ]
gap> N:=NG[2];
```

```
Group([ (1,3)(2,4) ])
gap> IsNormal(G,N);
true
gap> F:=FactorGroup(G,N);
Group([ f1 ])
gap> Size(F)=Size(G)/Size(N); Size(F); Size(G); Size(N);
true
2
4
2
```

Tanım : G bir grup ve $a,b \in G$ olsun. $a^{-1}b^{-1}ab$ elemanına a ve b nin komütatörü denir ve $[a,b]$ ile gösterilir [7].

GAP programında `Comm()` komutu iki elemanın komütatörünü hesaplamak için kullanılır. Buradaki `()` işaretleri arasına komütatörleri hesaplanacak olan elemanlar yazılmalıdır [17]. GAP programında daima sağ etki söz konusu olduğundan komütatör işlemini $a^{-1}b^{-1}ab$ biçiminde gerçekleştirir.

```
gap> a:=(1,3)(4,6);; b:=(1,6,5);;
gap> Comm(a,b);
(1,6,5,4,3)
```

Teorem : G grubunun değişmeli olması için gerek ve yeter koşul, $\forall a,b \in G$ olmak üzere

$$[a,b]=e$$

eşitliği sağlanır [7].

Bu önermeyi GAP programı kullanarak incelemek için abelyen olan bir grup alınmalı ve bu gruptaki tüm elemanların ikililer halinde seçilmesiyle elde edilen komütatörlerin birim eleman olduğu görülmelidir.

```
gap> G:=Group((1,2,3));
Group([ (1,2,3) ])
gap> IsAbelian(G);
true
```

```
gap> GE:=Elements(G);
[ (), (1,2,3), (1,3,2) ]
gap> Comm(GE[1],GE[2]); Comm(GE[1],GE[3]); Comm(GE[2],GE[3]);
()
()
()
```

Tanım : Bir grubun bütün komütatörlerinin ürettiği alt gruba komütatör alt grup denir ve $[G,G]$ ile gösterilir.

$$[G,G]=\{c_1c_2c_3...c_m : c_i \text{ komütatör}, i=1,2,...,m,\ m\in\mathbb{N}\}$$

dir [7].

`CommutatorSubgroup()` komutu aynı aileden iki grubun elemanlarının komütatörlerini eleman kabul eden bir grup oluşturur [17].

```
G:=Group((1,2,3));
Group([ (1,2,3) ])
gap> H:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
gap> K:=CommutatorSubgroup(G,H);
Group([ (1,4,2), (3,5,4) ])
gap> IsGroup(K);
true
```

Teorem : $[G,G]$, G grubunun nin bir normal alt grubudur [7].

```
gap> IsNormal(G,K);
true
gap> IsNormal(H,K);
true
```

Tanım : Bir G grubunun has (kendisi ve birim dışında), hiçbir normal alt grubu yoksa G grubuna basit grup denir [7].

```
gap> G:=Group((1,2,3));
Group([ (1,2,3) ])
gap> NormalSubgroups(G);
[ Group([ (1,2,3) ]), Group(()) ]
```

Örneğin yukarıdaki G grubunun kendisinden ve birimden farklı başka bir normal alt grubu olmadığından G grubu bir basit gruptur.

Tanım : G bir grup ve $M \lhd G$ olsun. M yi kapsayan, M ve G den başka hiçbir normal alt grubu yoksa , M ye G nin bir maksimal normal alt grubu denir [7]. Bir grubun maksimal normal alt grubu birden fazla olabilir.

`MaximalNormalSubgroups()` komutu bir grubun maksimal normal alt grubunu bulmak için kullanılır. Buradaki `()` işaretleri arasına maksimal normal alt grubu bulunacak olan grup yazılmalıdır [17].

```
gap> G:=Group((1,2,3,4,5,6));
Group([ (1,2,3,4,5,6) ])
gap> MaximalNormalSubgroups(G);
[ Group([ (1,3,5)(2,4,6) ]), Group([ (1,4)(2,5)(3,6) ]) ]
```

Teorem : $M \lhd G$ maksimal $\Leftrightarrow G/M$ basit olmasıdır [7].

Bu teoremi GAP programı kullanarak incelemek için bir G grubu alınmalı ve G grubunun bir maksimal normal alt grubu ile bölüm grubu oluşturulmalıdır. Oluşturulan bölüm grubunun kendisinden ve birimden başka normal alt grubu olmadığını yani, basit grup olduğu GAP programı aracılığı ile aşağıdaki gibi gösterilir.

```
gap> G:=Group((1,2,3,4,5,6,7,8));
Group([ (1,2,3,4,5,6,7,8) ])
gap> M:=MaximalNormalSubgroups(G);
[ Group([ (1,3,5,7)(2,4,6,8), (1,5)(2,6)(3,7)(4,8) ]) ]
gap> F:=FactorGroup(G,M[1]);
Group([ f1 ])
gap> U:=NormalSubgroups(F);
[ Group([ f1 ]), Group([ ]) ]
```

12 GRUP HOMOMORFIZMLERİ

Tanım : G ve G' iki grup olmak üzere $f: G \to G'$ fonksiyonu verilsin. Her $x, y \in G$ için

$$f(xy) = f(x)f(y)$$

ise f ye bir homomorfizm denir. Eğer $f(G) = G'$ ise G' ye G nin homomorfik görüntüsü denir [3].

Homomorfizm birebir ve örten olmadığından, G den G' ye bütün yapıyı taşımaz. Fakat işlem korur. Böylece homomorfizmin en önemli özelliği, G nin grup işlemlerini G' nün grup işlemlerine taşımasıdır [3].

G ve H birer grup olmak üzere GAP programında `GroupHomomorphismBy Images(G,H,[Gninüreteçleri],[bu üreteçlerin görüntülerinin listesi])` şeklindeki komut dizisi homomorfizmleri oluşturmak için kullanılır. Buradaki G grubuna, tanım (source) grubu ve H grubuna da değer (range) grubu denir [17].

Bu komut kullanılırken yazılan üreteçler G yi üretmezse veya üreteçlerin fonksiyonu bir homomorfizme genişletilemezse (örneğin üreteçlerin fonksiyonları sadece çok değişkenli fonksiyon tanımlıyorsa) `fail` sonucu elde edilecektir.

`GroupHomomorphismByImages` komutundan başka homomorfizm oluşturmak için `GroupHomomorphismByImagesNC` komutu da kullanılabilir. `NC` versiyonu üreteçlerin G yi üretip üretmediğini test etmez. Oluşturulan bir fonksiyonun grup homomorfizmi olup olmadığı `IsGroupHomomorphism()` komutu kullanılarak test edilebilir. `()` işaretleri arasına test edilmek istenilen fonksiyonun ismi yazılır. Elde edilen fonksiyon bir grup homomorfizmi ise GAP programında `true` yanıtı verilecektir [17].

```
gap> G:=Group((1,2,3)(4,5));
Group([ (1,2,3)(4,5) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3)(4,5) ]
gap> H:=Group((1,2)(3,4));
Group([ (1,2)(3,4) ])
gap> hom:=GroupHomomorphismByImages(G,H,genG,[(1,2)]);
[ (1,2,3)(4,5) ] -> [ (1,2) ]
gap> IsGroupHomomorphism(hom);
```

```
true
gap> a:=[(1,2)(4,5)];
[ (1,2)(4,5) ]
gap> a=genG;
false
gap> hom1:=GroupHomomorphismByImages(G,H,a,[(1,2)]);
fail
gap> hom1:=GroupHomomorphismByImagesNC(G,H,a,[(1,2)]);
[ (1,2)(4,5) ] -> [ (1,2) ]
gap> IsGroupHomomorphism(hom1);
true
```

GAP programında `Source()` komutu homomorfizmin tanım (source) grubunu sorgulamak `Range()` komutu da değer (range) grubunu sorgulamak için kullanılır [17].

```
gap> K:=Range(hom);
Group([ (1,2)(3,4) ])
gap> R:=Source(hom);
Group([ (1,2,3)(4,5) ])
gap> R=G;
true
gap> K=H;
true
```

`Image()` fonksiyonu tanım (source) grubundan alınan bir elemanın veya bir kümenin homomorfizm altındaki görüntüsünü bulmak için kullanılır [17].

```
gap> Elements(G);
[ (), (4,5), (1,2,3), (1,2,3)(4,5), (1,3,2), (1,3,2)(4,5) ]
gap> T:=[(4,5),(1,2,3),(1,3,2)];
[ (4,5), (1,2,3), (1,3,2) ]
gap> Image(hom,T);
[ (), (1,2) ]
gap> Image(hom,(4,5));
(1,2)
```

Örnek : G ve G' herhangi iki grup olsun. Bu durumda

$$f : G \to G'$$
$$x \to e$$

şeklinde tanımlanan fonksiyon bir homomorfizmdir. Çünkü

$$f(xy) = e = ee = f(x)f(y)$$

dir. Bu homomorfizme sıfır homomorfizmi denir [3].

```
gap> G:=Group((1,2,3));
Group([ (1,2,3) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3) ]
gap> H:=Group((1,3,2));
Group([ (1,3,2) ])
gap> eH:=Identity(H);
()
gap> hom2:=GroupHomomorphismByImages(G,H,genG,[eH]);
[ (1,2,3) ] -> [ () ]
gap> IsGroupHomomorphism(hom2);
true
gap> Elements(G);
[ (), (1,2,3), (1,3,2) ]
gap> Image(hom2,());
()
gap> Image(hom2,(1,2,3));
()
gap> Image(hom2,(1,3,2));
()
```

GAP programında `InverseGeneralMapping()` komutu bir fonksiyonun tersinin bulunmasında kullanılır. `InverseGeneralMapping()` komutu kullanılarak bir homomorfizmin tersi de bulunabilir. Buradaki `()` işaretleri arasına tersi bulunacak fonksiyon yazılmalıdır [17].

```
gap> InverseGeneralMapping(hom2);
[ () ] -> [ (1,2,3) ]
```

Teorem : $f:G\to G'$ grup homomorfizmi ve $e\in G$ birim eleman olmak üzere

i) $f(e)=e'$, G' nün birim elemanıdır.

ii) Her $x\in G$ için $f(x^{-1})=f(x)^{-1}$

özellikleri sağlanır [3].

Bu teoremi GAP programı kullanarak inceleyelim. Bunun için bir homomorfizm oluşturalım. Teoremin ilk kısmında homomorfizmin tanım (source) grubunun tüm elemanlarının homomorfizm altındaki görüntüsü değer (range) grubunun birim elemanı olacaktır.

```
gap> G:=Group((1,2,3),(1,2));
Group([ (1,2,3), (1,2) ])
gap> G1:=Group((1,3,2),(1,3));
Group([ (1,3,2), (1,3) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3), (1,2) ]
gap> hom3:=GroupHomomorphismByImages(G,G1,genG,[(),(1,2)]);
[ (1,2,3), (1,2) ] -> [ (), (1,2) ]
gap> eG:=Identity(G);
()
gap> eG1:=Identity(G1);
()
gap> Image(hom3,eG);
()
gap> gor:=Image(hom3,eG);
()
gap> gor=eG1;
true
```

Teoremin ikinci kısmını GAP programı kullanarak inceleyelim.

```
gap> ElemanG:=Elements(G);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
gap> e1:=Image(hom3,ElemanG[2]^-1);
(1,2)
gap> e2:=Image(hom3,ElemanG[2])^-1;
```

```
(1,2)
gap> e1=e2;
true
```

Bu eşitlik G grubunun tüm elemanları ile sağlanır. G grubundan başka elemanlar seçilerek bu eşitliğin sağlandığı gösterilebilir.

```
gap> e3:=Image(hom3,ElemanG[4]^-1);
()
gap> e4:=Image(hom3,ElemanG[4])^-1;
()
gap> e3=e4;
true
gap> e5:=Image(hom3,ElemanG[5]^-1);
()
gap> e6:=Image(hom3,ElemanG[5])^-1;
()
gap> e5=e6;
true
```

Teorem : $f : G \to H$ bir homomorfizma olsun.

i) G nin her alt grubunun f altındaki görüntüsü H nın bir alt grubudur.

ii) H nın her alt grubunun f altındaki ters görüntüsü G nin bir alt grubudur [7].

İlk önce teorimin ilk kısmını GAP programı kullanarak inceleyelim.

```
gap> G:=Group((1,2,3,4),(1,3,2));
Group([ (1,2,3,4), (1,3,2) ])
gap> H:=Group((1,4,2,3),(1,2,3));
Group([ (1,4,2,3), (1,2,3) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3,4), (1,3,2) ]
gap> hom4:=GroupHomomorphismByImages(G,H,genG,[(1,2),(1,2,3)]);
[ (1,2,3,4), (1,3,2) ] -> [ (1,2), (1,2,3) ]
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
```

```
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3) ]
gap> SG:=Subgroup(G,[(1,2,4,3)]);
Group([ (1,2,4,3) ])
gap> AH:=Image(hom4,SG);
Group([ (2,3) ])
gap> IsSubgroup(H,AH);
true
```

G grubunun alt gruplarından başka bir tanesi seçilerek bu teoremin sağlandığı tekrar görülebilir.

```
gap> S2G:=Subgroup(G,[(1,2,4)]);
Group([ (1,2,4) ])
gap> A2H:=Image(hom4,S2G);
Group([ (1,2,3) ])
gap> IsSubgroup(H,A2H);
true
```

Teoremin ikinci kısmını GAP programı kullanarak inceleyelim.

```
gap> hom5:=InverseGeneralMapping(hom4);
[ (1,2), (1,2,3) ] -> [ (1,2,3,4), (1,3,2) ]
gap> eH:=Elements(H);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3) ]
gap> BH:=Subgroup(H,[(2,3)]);
Group([ (2,3) ])
gap> K:=Image(hom5,BH);
Group([ (1,4)(2,3), (1,2)(3,4), (1,2,4,3) ])
gap> IsSubgroup(G,K);
true
```

H grubunun alt gruplarından başka bir tanesi seçilerek bu teoremin sağlandığı tekrar gösterilebilir.

```
gap> B2H:=Subgroup(H,[(1,3,2)]);
Group([ (1,3,2) ])
gap> L:=Image(hom5,B2H);
Group([ (1,4)(2,3), (1,2)(3,4), (1,2,3) ])
gap> IsSubgroup(G,L);
true
```

GAP programında bir homomorfizmin birebir olup olmadığı `IsInjective()` komutu ile test edilir. `()` işaretleri arasına birebir olup olmadığı test edilecek olan fonksiyon yazılmalıdır [17].

GAP programında bir homomorfizmin örten olup olmadığı `IsSurjective()` komutu ile test edilir. `()` işaretleri arasına örten olup olmadığı test edilecek olan fonksiyon yazılmalıdır [17].

GAP programında bir homomorfizmin hem birebir hem de örten olup olmadığını ise `IsBijective()` komutu ile test edilir. Bu sorular `()` işaretleri arasında kullanılacak homomorfizmin özelliğine göre `true` veya `false` şeklinde yanıt bulacaktır [17].

```
gap> IsInjective(hom4);
false
gap> IsSurjective(hom4);
false
gap> IsBijective(hom4);
false
```

Teorem : $f:G\to H$ bir homomorfizma olsun.

i) f örten ise G nin her normal alt grubunun f altındaki görüntüsü H nın bir normal alt grubudur.

ii) H nın her normal alt grubunun f altındaki ters görüntüsü G nin bir normal alt grubudur [7].

Teoremin ilk kısmını GAP programı kullanarak inceleyelim. Bunun için örten olacak şekilde bir homomorfizm oluşturalım.

```
gap> G:=Group((1,2,3,4),(1,3));
Group([ (1,2,3,4), (1,3) ])
gap> H:=Group((1,4,6,7)(2,3,5,8),(1,5)(2,6)(3,4)(7,8));
Group([ (1,4,6,7)(2,3,5,8), (1,5)(2,6)(3,4)(7,8) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3,4), (1,3) ]
gap> genH:=GeneratorsOfGroup(H);
[ (1,4,6,7)(2,3,5,8), (1,5)(2,6)(3,4)(7,8) ]
gap> hom6:=GroupHomomorphismByImages(G,H,genG,genH);
[(1,2,3,4),(1,3)] -> [(1,4,6,7)(2,3,5,8), (1,5)(2,6)(3,4)(7,8)]
gap> IsSurjective(hom6);
true
gap> NG:=NormalSubgroups(G);
[Group([(1,2,3,4),(1,3)]),Group([(2,4),(1,3)(2,4)]),
Group([(1,2)(3,4),(1,3)(2,4)]),Group([(1,2,3,4),(1,3)(2,4)]),
Group([(1,3)(2,4)]),Group(()) ]
gap> K:=NG[2];
Group([ (2,4), (1,3)(2,4) ])
gap> U:=Image(hom6,K);
Group([ (1,2)(3,7)(4,8)(5,6), (1,6)(2,5)(3,8)(4,7) ])
gap> IsSubgroup(H,U);
true
```

G grubunun normal alt gruplarından başka bir tanesi seçilerek bu teoremin sağlandığı görülebilir.

```
gap> M:=NG[4];
Group([ (1,2,3,4), (1,3)(2,4) ])
gap> Y:=Image(hom6,M);
Group([ (1,4,6,7)(2,3,5,8), (1,6)(2,5)(3,8)(4,7) ])
Group([ (1,4,2,3), (1,2,3) ])
gap> IsSubgroup(H,Y);
true
```

Teoremin ikinci kısmını GAP programı kullanarak inceleyelim.

```
gap> hom7:=InverseGeneralMapping(hom6);
[(1,4,6,7)(2,3,5,8),(1,5)(2,6)(3,4)(7,8) ] -> [(1,2,3,4),(1,3)]
gap> NH:=NormalSubgroups(H);
[Group([(1,4,6,7)(2,3,5,8),(1,5)(2,6)(3,4)(7,8)]),
Group([ (1,5)(2,6)(3,4)(7,8),(1,6)(2,5)(3,8)(4,7)]),
Group([ (1,8)(2,7)(3,6)(4,5),(1,6)(2,5)(3,8)(4,7)]),
Group([ (1,4,6,7)(2,3,5,8),(1,6)(2,5)(3,8)(4,7)]),
Group([ (1,6)(2,5)(3,8)(4,7) ]),Group(())]
gap> V:=NH[4];
Group([ (1,4,6,7)(2,3,5,8), (1,6)(2,5)(3,8)(4,7) ])
gap> L:=Image(hom7,V);
Group([ (1,2,3,4), (1,3)(2,4) ])
gap> IsNormal(G,L);
true
```

H grubunun normal alt gruplarından başka bir tanesi de seçilerek bu teoremin sağlandığı tekrar görülebilir.

```
gap> W:=NH[2];
Group([ (1,5)(2,6)(3,4)(7,8), (1,6)(2,5)(3,8)(4,7) ])
gap> T:=Image(hom7,W);
Group([ (1,3), (1,3)(2,4) ])
gap> IsNormal(G,T);
true
```

Tanım : $f: G \to G'$ bir homomorfizm olsun.

$$Çekf = \{x \in G : f(x) = e_{G'}\}$$

şeklinde tanımlanan kümeye f nin **çekirdeği** denir [3].

Tanım : $f: G \to G'$ bir homomorfizm olsun.

$$Görf = \{y \in G' : x \in G \text{ için } y = f(x)\}$$

kümesine ise f nin **görüntü kümesi** denir [3].

GAP programında `Kernel()` komutu homomorfizmin çekirdeğini `Image()` komutu ise homomorfizmin görüntü kümesini bulmak için kullanılır [17].

```
gap> Kernel(hom4);
Group([ (1,4)(2,3), (1,2)(3,4) ])
gap> Image(hom4);
Group([ (1,2), (1,2,3) ])
gap> Kernel(hom6);
Group(())
gap> Image(hom6);
Group([ (1,4,6,7)(2,3,5,8), (1,5)(2,6)(3,4)(7,8) ])
```

Teorem : Eğer bir $f : G \to H$ grup homomorfizmi birebir ise ancak ve ancak

$$Çekf = \{e_G\}$$

olması gerekir [8].

Bu teoremi GAP programı kullanarak incelemek için birebir bir homomorfizm oluşturalım ve $Çekf = \{e_G\}$ olduğunu görelim.

```
gap> G:=Group((1,2,3)(4,5));
Group([ (1,2,3)(4,5) ])
gap> genG:=GeneratorsOfGroup(G);
[ (1,2,3)(4,5) ]
gap> hom8:=GroupHomomorphismByImages(G,G,genG,genG);
[ (1,2,3)(4,5) ] -> [ (1,2,3)(4,5) ]
gap> IsInjective(hom8);
true
gap> Kernel(hom8);
Group(())
```

13 SİMETRİK GRUPLAR

n elemanlı bir kümenin kendi üzerine birebir bir fonksiyonuna bir n-li permütasyon denir. Bütün n-li permütasyonların kümesinin, bileşke işlemi altında bir grup oluşturduğunu biliyoruz. Bu grup S_n ile gösterilir ve simetrik grup veya permütasyon grubu olarak adlandırılır.

$f \in S_n$ ve $f, \{x_1, x_2, \ldots, x_n\}$ kümesinin kendi üzerine bir birebir fonksiyon olsun. $i_1, i_2, \ldots, i_k$: $1, 2, \ldots, n$ nin bir değişik sırada sıralanışı, yani bir permütasyon olmak üzere, $f(x_k) = x_i$ $(k = 1, 2, \ldots, n)$ ise f fonksiyonunu, her elemanın altına görüntüsünü yazarak;

$$f = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \\ x_{i_1} & x_{i_2} & \ldots & x_{i_n} \end{pmatrix} = \begin{pmatrix} x_k \\ x_{i_k} \end{pmatrix}$$

ile veya x leri yazmayarak;

$$f = \begin{pmatrix} 1 & 2 & \ldots & n \\ i_1 & i_2 & \ldots & i_n \end{pmatrix}$$

ile gösterilir [7].

Tanım : Bir f permütasyonu, $j_1, j_2, \ldots j_k$ $(k > 1)$ farklı doğal sayılar olmak üzere ;

$$f(j_1) = j_2, f(j_2) = j_3, \ \ldots, f(j_{k-1}) = j_k, f(j_k) = j_1$$

ile tanımlı ise $f = (j_1 j_2 \ldots j_k)$ ile gösterilir ve k uzunluğunda bir devir denir. 1 uzunluğundaki bir devir de özdeşlik fonksiyonu olarak adlandırılır.

Bir deviri, saat yönünde yönlendirilmiş çember üzerinde dizilmiş semboller olarak düşünebilir ve herhangi bir j den başlayarak;

$$f = (j_1 j_2 \ldots j_k) = (j_2 j_3 \ldots j_k j_1) = \ldots = (j_k j_1 \ldots j_{k-1})$$

farklı şekillerde yazabiliriz [7].

GAP programında `IsPerm()` komutu ile devirlerin permütasyon olup olmadığı test edilebilir [17].

```
gap> a:=(1,2,3,4,5);
(1,2,3,4,5)
gap> b:=(2,3,4,5,1);
(1,2,3,4,5)
gap> a=b;
```

```
true
gap> c:=(4,5,1,2,3);
(1,2,3,4,5)
gap> a=c;
true
gap> b=c;
true
gap> k:=(1,3,5,4);
(1,3,5,4)
gap> IsPerm(k);
true
```

Tanım : İki devir ortak eleman bulundurmuyorsa bu devirlere ayrık devirler denir [4].

Tanım : Genel olarak iki permütasyonun çarpımı (bileşkesi) değişmeli değildir. Ayrık devirlerin çarpımı (bileşkesi) değişmelidir [7].

```
gap> a:=(1,2,3,4);
(1,2,3,4)
gap> b:=(1,5);
(1,5)
gap> ab:=a*b;
(1,2,3,4,5)
gap> ba:=b*a;
(1,5,2,3,4)
gap> ab=ba;
false
gap> x:=(1,2,3,4);
(1,2,3,4)
gap> y:=(5,6,7,8);
(5,6,7,8)
gap> xy:=(1,2,3,4)*(5,6,7,8);
(1,2,3,4)(5,6,7,8)
gap> yx:=(5,6,7,8)*(1,2,3,4);
(1,2,3,4)(5,6,7,8)
gap> xy=yx;
```

```
true
```

Teorem : r uzunluğundaki bir devrin mertebesi r dir [7].

Bu teoremi GAP programı kullanarak inceleyelim. GAP programında `Order()` komutu bir devrin mertebesini bulmak için kullanılır [17].

```
gap> A:=(1,2,3,4,5,6,7);
(1,2,3,4,5,6,7)
gap> B:=(1,5,7);
(1,5,7)
gap> Order(A);
7
gap> Order(B);
3
```

Teorem : S_n deki her permütasyon sıra gözetmeksizin, ayrık devirlerin çarpımı olarak tek türlü yazılabilir [7].

Örnek :

$f=\begin{pmatrix}1&2&3&4&5&6\\1&5&4&6&2&3\end{pmatrix}=(1)(2\ 5)(3\ 4\ 6)$ dır. 1 uzunluğundaki devirler yazılmayabilir [7].

Teorem : Bir $f\in S_n$ permütasyon unun mertebesi, ayrıldığı ayrık devirlerin uzunluklarının ortak katlarının en küçüğüdür [7].

Örnek : $f=(3\ 4)(1\ 2\ 5)$ permütasyonu için , $|f|=\text{OKEK}(2,3)=6$ ve ayrık devirlerden oluşmayan $g=(3\ 4)(1\ 2\ 3\ 5)=(1\ 2\ 3\ 4\ 5)$ permütasyonu ayrık devirlerden oluşmayıp $|f|=5$ dir [7].

```
gap> a:=(3,4);
(3,4)
gap> b:=(1,2,5);
(1,2,5)
gap> f:=(3,4)(1,2,5);
(1,2,5)(3,4)
```

```
gap> Order(f);
6
gap> Order(a);
2
gap> Order(b);
3
gap> Lcm(2,3);
6
gap> c:=(3,4);
(3,4)
gap> d:=(1,2,3,5);
(1,2,3,5)
gap> g:=(1,2,3,4,5);
(1,2,3,4,5)
gap> Order(c);
2
gap> Order(d);
4
gap> Order(g);
5
gap> Lcm(2,4);
4
```

Teorem : Her devir ikili devirlerin bir çarpımıdır [7].

Örnek : $f = (1\ 2\ 3\ 4) = (1\ 4)(1\ 3)(1\ 2)$ [7]

```
gap> g:=(1,2,3,4,5);
(1,2,3,4,5)
gap> h:=(1,2)*(1,3)*(1,4)*(1,5);
(1,2,3,4,5)
gap> g=h;
true
```

Tanım : Bir permütasyon çift sayıda 2 linin çarpımı ise bu permütasyona çift, aksi halde tek permütasyon denir [7].

GAP programında `SignPerm()` komutu bir permütasyonun tek mi yoksa çift mi olduğunu belirmek için kullanılır. 2 lilerin sayısı k olmak üzere bu komut $(-1)^k$ biçiminde işlev görür. `SignPerm()` komutu k sayısı çift yani permütasyon çift ise 1, k sayısı tek yani permütasyon tek ise -1 yanıtını verecektir [17].

```
gap> k:=(1,2,3,4,5,6);
(1,2,3,4,5,6)
gap> l:=(1,2)*(1,3)*(1,4)*(1,5)*(1,6);
(1,2,3,4,5,6)
gap> SignPerm(l);
-1
gap> SignPerm(k);
-1
gap> SignPerm(h);
1
```

Tanım : G bir grup ve $a,b \in G$ olsun. $b = xax^{-1}$ olacak şekilde bir $x \in G$ varsa b ye a nın bir eşleniği denir [7].

Teorem : $\sigma \in S_n$ için ,

$$\sigma(i_1 i_2 \ldots i_r)\sigma^{-1} = (\sigma(i_1)\sigma(i_2)\ldots\sigma(i_r))$$

eşitliği sağlanır [7].

Burada $\sigma(i_1 i_2 \ldots i_r)\sigma^{-1}$ işlemi bileşke işlemi olduğundan bileşke bulunurken işlemin sondan başlayarak yapılması gerektiğine ve GAP programında tüm işlemlerin soldan sağa doğru yapılması gerektiğine dikkat edilmelidir.

Örnek : S_4 de $\sigma = (1\ 3\ 2)$ olmak üzere,

$$\sigma(1\ 2\ 4\ 3)\sigma^{-1} = (\sigma(1)\sigma(2)\sigma(4)\sigma(3)) = (3\ 1\ 4\ 2)$$

eşitliğinin sağlandığını gösterelim [7].

```
gap> s4:=SymmetricGroup(IsPermGroup,4);
Sym( [ 1 .. 4 ] )
gap> es4:=Elements(s4);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
```

```
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> sigma:=(1,3,2);
(1,3,2)
gap> sigma in es4;
true
gap> (1,2,4,3) in es4;
true
gap> c1:=sigma^-1*(1,2,4,3)*sigma;
(1,4,2,3)
gap> c2:=(1^sigma,2^sigma,4^sigma,3^sigma);
(1,4,2,3)
gap> c1=c2;
true
gap> (1,4,2,3)=(3,1,4,2);
true
```

Örnek :

$$f=\begin{pmatrix}1&2&3&4&5\\3&5&4&1&2\end{pmatrix} \text{ nin } \begin{pmatrix}1&2&3&4&5\\3&4&1&5&2\end{pmatrix}$$

ile eşleniğini bulalım.

$f=(1\ 3\ 4)(2\ 5)$ ve $\sigma=(1\ 3)(2\ 4\ 5)$ olduğuna göre ;

$$\begin{aligned}\sigma f\sigma^{-1}&=\sigma(134)(25)\\&=(\sigma(1\ 3\ 4)\sigma^{-1})(\sigma(2\ 5)\sigma^{-1})\\&=(\sigma(1)\sigma(3)\sigma(4))(\sigma(2)\sigma(5))\\&=(3\ 1\ 5)(4\ 2)\end{aligned}$$

bulunur [7].

Yukarıdaki örneği GAP programı kullanarak inceleyelim

```
gap> f:=(1,3,4)(2,5);
(1,3,4)(2,5)
gap> sigma:=(1,3)(2,4,5);
(1,3)(2,4,5)
```

```
gap> e1:=sigma^-1*f*sigma;
(1,5,3)(2,4)
gap> e2:=(1^sigma,3^sigma,4^sigma)(2^sigma,5^sigma);
(1,5,3)(2,4)
gap> e1=e2;
true
```

Tanım : $n \in \mathbb{N}$ olsun. $n_1 \leq n_2 \leq \ldots n_r$ ve $n = n_1 + n_2 + \ldots n_r$ koşullarını sağlayan bir $n_1, n_2, \ldots n_r$ doğal sayılar dizisine n nin bir ayrışımı denir. n nin tüm ayrışımları sayısı $p(n)$ ile gösterilir [7].

GAP programında bir n sayısının ayrışımını bulmak için `Partitions()` komutu kullanılır. Buradaki `()` işaretleri arasına ayrışımı bulunacak olan n doğal sayısı yazılmalıdır. `NrPartitions()` komutu n nin tüm ayrışımlarının sayısını verecektir [17].

Örnek : Bazı n değerleri için ayrışım örnekleri aşağıda verilmiştir [7].

$$1 = 1 \Rightarrow p(1) = 1$$
$$2 = 2 = 1+1 \Rightarrow p(2) = 2$$
$$3 = 3 = 1+2 = 1+1+1 \Rightarrow p(3) = 3$$
$$4 = 4 = 1+3 = 1+1+2 = 1+1+1+1 = 2+2 \Rightarrow p(4) = 5$$

```
gap> Partitions(1);
[ [ 1 ] ]
gap> NrPartitions(1);
1
gap> Partitions(2);
[ [ 1, 1 ], [ 2 ] ]
gap> NrPartitions(2);
2
gap> Partitions(3);
[ [ 1, 1, 1 ], [ 2, 1 ], [ 3 ] ]
gap> NrPartitions(3);
3
gap> Partitions(4);
[ [ 1, 1, 1, 1 ], [ 2, 1, 1 ], [ 2, 2 ], [ 3, 1 ], [ 4 ] ]
```

```
gap> NrPartitions(4);
5
gap> Partitions(7);
[[ 1, 1, 1, 1, 1, 1, 1 ],[ 2, 1, 1, 1, 1, 1 ],[ 2, 2, 1, 1, 1 ],
[ 2, 2, 2, 1 ],[ 3, 1, 1, 1, 1 ],[ 3, 2, 1, 1 ],[ 3, 2, 2 ],
[ 3, 3, 1 ],[ 4, 1, 1, 1 ],[ 4, 2, 1 ],[ 4, 3 ],[ 5, 1, 1 ],
[ 5, 2 ],[ 6, 1 ],[ 7 ]]
gap> NrPartitions(7);
15
gap> NrPartitions(100);
190569292
```

$n=100$ değeri için n nin tüm ayrışımlarının sayısı 190569292 olup `Partitions(100)` komutu kullanılarak bu ayrışımlar bulunabilir. Kuvvetli bir bilgisayar konfigürasyonuna sahip olunmazsa bu ayrışımları hesaplarken bilgisayarı uzun süre yanıt vermemesi gayet doğaldır.

Teorem : S_n de eşlenik sınıflarının sayısı $p(n)$ dir [7].

Örnek : S_3 ün eşlenik sınıflarını

$$C_1=\{I\}\ ,\ C_2=\{(1\ 2),(1\ 3),(2\ 3)\}\ C_3=\{(1\ 2\ 3),(1\ 3\ 2)\}$$

olup üç tanedir [7].

```
gap> NrPartitions(3);
3
```

Örnek : S_6 da , $a=(1\ 2\ 3\ 4\ 5\ 6)\in S_6$ için $M(a)$ merkezleştiricisini bulalım ve $M(a)$ nın S_6 nın bir alt grubu olduğunu gösterelim.

```
gap> S5:=SymmetricGroup(IsPermGroup,5);
Sym( [ 1 .. 5 ] )
gap> a:=(1,2,3)(4,5);
(1,2,3)(4,5)
gap> a in S6;
true
```

```
gap> Ma:=Centralizer(G,a);
Group([ (1,2,3) ])
gap> IsSubgroup(S6,Ma);
true
```

14 SYLOW TEOREMLERİ

Lagrange teoremine göre, bir sonlu grubun her alt grubunun mertebesinin grubun mertebesini böldüğünü biliyoruz. Fakat bu teoremin karşıtı doğru olmayabilir [7].

Teorem : G değişmeli ve sonlu bir grup olsun. p asal ve $p \mid |G|$ ise G de mertebesi p olan bir eleman, dolayısı ile G nin p. mertebeden bir alt grubu vardır [7].

Bu teoremin doğruluğunu GAP programı kullanarak inceleyelim.

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> eG:=Elements(G);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> Size(G);
4
gap> p:=2;
2
gap> Size(G)/p;
2
gap> eG:=Elements(G);
[ (), (1,2,3,4), (1,3)(2,4), (1,4,3,2) ]
gap> Order(eG[1]);
1
gap> Order(eG[2]);
4
gap> Order(eG[3]);
2
gap> Order(eG[4]);
4
gap> U:=Subgroup(G,[eG[3]]);
Group([ (1,3)(2,4) ])
gap> IsSubgroup(G,U);
true
gap> Size(U);
2
```

Tanım : $|G| = mp^k$, p asal ve $p \nmid m$ ise p^k mertebeden bir alt gruba G nin **p-Sylow alt grubu** denir [7].

GAP programında `SylowSubgroup()` komutu bir grubun p-Sylow alt grubunu bulmak için kullanılır. `()` işaretleri arasına sırasıyla p-Sylow alt grubu bulunacak olan grup ve p asal sayısı yazılmalıdır. `SylowSubgroup()` p sayısını seçiminin, p asal $m \in \mathbb{N}$, $p \nmid m$ ve $|G| = mp^k$ olup olmadığını test etmez. p sayısının seçimi bu şartları sağlamıyorsa `SylowSubgroup()` komutu yalnızca birim elemandan oluşan sylow alt grubu verecektir [17].

```
gap> G:=Group((1,2,3,4,5,6,7,8,9,10,11,12));
Group([ (1,2,3,4,5,6,7,8,9,10,11,12) ])
gap> Size(G);
12
gap> p:=3;
3
gap> k:=1;
1
gap> m:=Size(G)/(p^k);
4
gap> SG:=SylowSubgroup(G,p);
Group([ (1,5,9)(2,6,10)(3,7,11)(4,8,12) ])
gap> G1:=Group((1,2,3,4,5,6,7,8,9,10));
Group([ (1,2,3,4,5,6,7,8,9,10) ])
gap> Size(G1);
10
gap> k1:=3;
3
gap> p1:=3;
3
gap> m1:=Size(G1)/(p1^k1);
10/27
gap> SG1:=SylowSubgroup(G1,p1);
Group(())
```

```
gap> p2:=13;
13
gap> SG2:=SylowSubgroup(G1,p2);
Group(())
```

Sylow Teoremi : p bir asal sayı ve $k \geq 0$, $p^k \mid |G|$ ise G grubunun , p^k mertebeden bir alt grubu vardır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> G:=SymmetricGroup(IsPermGroup,4);
Sym( [ 1 .. 4 ] )
gap> Size(G);
24
gap> p:=2;
2
gap> k:=3;
3
gap> Size(G)/p^k;
3
gap> PG:=SylowSubgroup(G,p);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> Order(PG)=p^k;
true
gap> IsSubgroup(G,PG);
true
```

Teorem : P , G nin bir p-sylow alt grubu, $a \in G$ ve $|a|$, p asal sayısının bir kuvveti olsun. Bu takdirde $aPa^{-1} = P$ ise $a \in P$ dir [7].

Bu teoremi GAP programı kullanarak inceleyelim. P nin G içindeki normalleştiricisi tanımından $aPa^{-1} = P \Leftrightarrow a \in N(P)$ olur. Bu durumda $|a|$ sayısının p asal sayısının bir katı olduğu durumda $a \in N(P)$ olduğu gösterilirse $a \in P$ olduğu gösterilmiş olur.

```
gap> G:=SymmetricGroup(IsPermGroup,4);
Sym( [ 1 .. 4 ] )
gap> p:=2;
2
gap> P:=SylowSubgroup(G,p);
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> eG:=Elements(G);
[(),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
(1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
(1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
(1,4)(2,3)]
gap> a:=eG[23];
(1,4,2,3)
gap> Order(a);
4
gap> Order(a)/p;
2
gap> P:=SylowSubgroup(G,p)
Group([ (1,2), (3,4), (1,3)(2,4) ])
gap> IsSubgroup(G,P);
true
gap> NP:=Normalizer(G,P
Group([ (1,4)(2,3), (3,4), (1,2) ])
gap> eNP:=Elements(NP);
[ (), (3,4), (1,2), (1,2)(3,4), (1,3)(2,4), (1,3,2,4),
(1,4,2,3), (1,4)(2,3) ]
gap> a in eNP;
true
gap> a in P;
true
```

Örnek : Mertebesi 42 olan bir grubun 7-Sylow alt grubunun bir normal alt grup olduğunu gösterelim [7].

```
gap> G:=Group((1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
```

```
20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,4
1,42));
Group([(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22
,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42)])
gap> Size(G);
42
gap> SG:=SylowSubgroup(G,7);
Group([(1,7,13,19,25,31,37)(2,8,14,20,26,32,38)(3,9,15,21,27,33,
39)(4,10,16,22,28,34,40)(5,11,17,23,29,35,41)(6,12,18,24,30,36,4
2)])
gap> IsNormal(G,SG);
true
```

15 HALKALAR

Tanım : Bir H kümesi üzerinde

$$+ : H \times H \to H \qquad \cdot : H \times H \to H$$
$$(x, y) \mapsto x + y \quad \text{ve} \quad (x, y) \mapsto x \cdot y$$

ikili işlemleri verilsin.

H1. $(H, +)$ bir abelyen grup,

H2. $\forall x, y, z \in H$ için $x \cdot (y \cdot z) = (x \cdot y) \cdot z$,

H3. $\forall x, y, z \in H$ için

$$x \cdot (y + z) = x \cdot y + x \cdot z \text{ ve } (x + y) \cdot z = x \cdot z + y \cdot z$$

bu üç özellik sağlanıyorsa, H kümesine bir halka denir ve $(H, +, \cdot)$ ile gösterilir [3].

H4. H bir halka olsun. Her $x \in H$ için

$$x \cdot 1 = 1 \cdot x = x$$

olacak şekilde $1 \in H$ elemanı varsa H halkasına birimli halka denir ve bu eleman 1_H veya 1 ile gösterilir [3].

H5. Bir H halkasındaki her $x, y \in H$ için $x \cdot y = y \cdot x$ ise H ya değişmeli halka denir [3].

Gösterim kolaylığı sağlaması bakımından yukarıda verilen ikili işlemler "+" ve "·" olarak alınmıştır. $(H, +)$ abelyen grup olduğundan H ın toplamsal birimi olan 0_H elemanına sıfır eleman denir. $(H2)$ aksiyomuna birleşme ve $(H3)$ aksiyomuna soldan ve sağdan dağılma özelliği denir [3].

GAP programında `Ring()` ve `DefaultRing()` komutları halka yapısı oluşturmak için kullanılır. `()` işaretleri arasına bu halka içersinde yer alacak elemanlardan bir veya birkaçı sırayla veya liste halinde yazılmalıdır [17].

```
gap> Ring(1,-1);
Integers
gap> DefaultRing(1,-1);
Integers
gap> Ring([1..5]);
Integers
```

```
gap> DefaultRing([1..5]);
Integers
```

`Ring()` ve `DefaultRing()` komutları arasındaki temel fark `Ring()` komutunda verilen elemanları içeren en küçük halkanının bulunabilmesi `DefaultRing()` komutunda ise verilen elemanları içeren daha büyük halkaların bulunabilmesidir. GAP programında `IsRing()` komutu bir cebirsel yapının halka olup olmadığını bulmak için kullanılır [17].

```
gap> H:=Ring(2,-2);
<ring with 2 generators>
gap> G:=DefaultRing(2,-2);
Integers
gap> IsRing(G);
true
```

GAP programında `GeneratorsOfRing()` komutu bir halkanın üreteçlerinin bulunması için kulalnılır. `()` işaretleri arasına üreteçleri bulunmak istenilen halka yazılmalıdır.

```
gap> GeneratorsOfRing(H);
[ 2, -2 ]
gap> GeneratorsOfRing(G);
[ 1 ]
```

GAP programında soldan dağılma ($x\cdot(y+z)=x\cdot y+x\cdot z$) özelliğinin olup olmadığı `IsLDistributive()` komutu kullanılarak, sağdan dağılma ($(x+y)\cdot z=x\cdot z+y\cdot z$) özelliğinin olup olmadığı `IsRDistributive()` komutu kullanılarak bulunur. Bu iki kıyaslama da doğru ise dağılma özelliği sağlanır. GAP programında dağılma özelliğinin sağlanıp sağlanmadığı `IsDistributive()` komutu kullanılarak bulunur [17].

```
gap> H:=Ring(2,0);
<ring with 2 generators>
gap> IsRing(H);
true
gap> IsLDistributive(H);
true
```

```
gap> IsRDistributive(H);
true
gap> IsDistributive(H);
true
```

GAP programında `Integrals`, `PositiveIntegrals` ve `Rationals` komutları tam sayılar, pozitif tamsayılar ve rasyonel sayılar kümesini ifade etmek için kullanılırlar. Bilinen çarpma ve toplama işlemlerine göre $\mathbb{Z}$ ve $\mathbb{Q}$ birimli ve değişmeli birer halkadır. $2\mathbb{Z}$ çift tamsayılar kümesi de birimi olmayan değişmeli bir halkadır. GAP programında `One()` komutu halkanın birim elemanını, `Zero()` komutu ise halkanın sıfır elemanını bulmak için kullanılır. `IsCommutative()` veya `IsAbelian()` komutları bir halkanın değişmeli olup olmadığını belirlemek için kullanılır [17].

```
gap> ZZ:=Integers;
Integers
gap> IsRing(ZZ);
true
gap> Zero(ZZ);
0
gap> One(ZZ);
1
gap> IsCommutative(ZZ);
true
gap> Q:=Rationals;
Rationals
gap> IsRing(Q);
true
gap> Zero(Q);
0
gap> One(Q);
1
gap> IsCommutative(Q);
true
gap> IsRing(2*ZZ);
true
```

```
gap> IsAbelian(2*ZZ);
true
gap> Zero(2*ZZ);
0
gap> One(2*ZZ);
Error, no method found! For debugging hints type ?Recovery from
NoMethodFound
Error, no 1st choice method found for `Representative' on 1
arguments called f\
rom
Representative( M ) called from
<function>( <arguments> ) called from read-eval-loop
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

Yukarıdaki örnekte $2\mathbb{Z}$ halkası birimli olmadığından birimi sorulduğunda GAP programı hata verir [17].

$\mathbb{Z}_n$ kümesi $\bmod n$ ye göre toplama ve çarpma işlemleri ile birlikte bir halkadır. n sayısının p gibi bir asal sayı olduğunda $\mathbb{Z}_p$ bir birimli ve değişmeli halkadır [3].

GAP programında $\mathbb{Z}_p$ şeklinde bir kümeyi tanımlamak için `Z()` komutu kullanılır. `()` işaretleri arasına p bir asal sayı ve n herhangi bir tamsayı olmak üzere p^n özelliğini sağlayabilecek nitelikteki tam sayılar yazılabilir. `Z()` komutu ile p^n özelliğini sağlamayan bir sayı kullanılırsa GAP programı hata verecektir [17].

```
gap> H:=Ring(Z(7));
GF(7)
gap> Elements(H);
[ 0*Z(7), Z(7)^0, Z(7), Z(7)^2, Z(7)^3, Z(7)^4, Z(7)^5 ]
gap> IsAbelian(H);
true
```

Yukarıdaki örnekte `0*Z(7)` elemanı halkanın sıfır elemanını `Z(7)^0` elemanı ise halkanın birim elemanı göstermektedir.

```
gap> 0*Z(7)+Z(7)^2;
Z(7)^2
gap> Z(7)^0*Z(7)^2;
Z(7)^2
gap> One(H);
Z(7)^0
gap> Zero(H);
0*Z(7)
```

GAP programında $\mathbb{Z}_p$ şeklinde bir kümeyi tanımlamak için `Z()` komutunun kullanılabileceğinden bahsedilmişti. GAP programında `ZmodnZ()` komutu da mod n ye göre toplama ve çarpma işlemleri ile birlikte bir halkayı tanımlamak için kullanılır. `Z()` ve `ZmodnZ()` komutları arasında iki temel fark vardır. Bunlardan birincisi `Z()` komutu ile elde edilen kümeyi bir halka haline getirmek için `Ring()` (veya `DefaultRing()`) komutunun kullanılması gerekirken `ZmodnZ()` komutu direkt olarak halka tanımlar. Bu iki komut arasındaki ikinci ve önemli fark ise `Z()` komutunda `()` işaretleri arasına p bir asal sayı ve n herhangi bir tamsayı olmak üzere p^n özelliğini sağlayabilecek nitelikteki tam sayılar yazılabilirken `ZmodnZ()` komutundaki `()` işaretleri arasına herhangi bir pozitif tamsayı yazılabilir [17].

```
gap> Ring(Z(5))=ZmodnZ(5);
true
gap> H:=ZmodnZ(4);
(Integers mod 4)
gap> IsRing(H);
true
gap> Elements(H);
[ZmodnZObj(0,4),ZmodnZObj(1,4),ZmodnZObj(2,4),ZmodnZObj(3,4 )]
```

Buradaki `ZmodnZObj(0,4)` ifadesi tam sayılar kümesinde mod 4 e göre 0 sayısına denk olan sayıları gösterir.

Tamsayılar için geçerli olan aritmetik kurallar herhangi bir halka için geçerli olmayabilir. Örneğin

$$a^2 - b^2 = (a+b)(a-b)$$

eşitliği halkanın değişmeli olmasına bağlıdır. Çünkü genel olarak $ab \neq ba$ olduğundan $-ab + ba \neq 0$ dır. Bununla birlikte bazı formüller kolayca elde edilir [3].

Teorem : H bir halka olsun. Bu durumda

i) $\forall x \in H$ için $x0 = 0x = 0$,

ii) $\forall x, y \in H$ için $(-x)y = x(-y) = -(xy)$,

iii) $\forall x, y \in H$ için $(-x)(-y) = xy$,

iv) $\forall x, y, z \in H$ için $x(y-z) = xy - xz$ ve $(y-z)x = yx - zx$,

v) $\forall x \in H$ için $(-1)x = -x$

vi) $(-1)(-1) = 1$

özellikleri sağlanır [3].

Bu teoremi GAP programı kullanarak incelemek için bir halka oluşturulabilir. Bu halka elemanları için bu özelliklerin sağlandığını birer örnek ve `ForAll()` komutu kullanarak görelim.

```
gap> H:=Ring(Z(13));
GF(13)
gap> eH:=Elements(H);
[0*Z(13),Z(13)^0,Z(13),Z(13)^2,Z(13)^3,Z(13)^4,Z(13)^5,
Z(13)^6,Z(13)^7,Z(13)^8,Z(13)^9,Z(13)^10,Z(13)^11]
gap> Zero(H);
0*Z(13)
gap> eH[7]*Zero(H)=Zero(H);
true
gap> ForAll(H,i -> i*Zero(H)=Zero(H));
true
gap> (-eH[7])*eH[8]=-(eH[7]*eH[8]);
true
gap> ForAll(H,x -> ForAll(H,y -> (-x)*y=-(x*y)));
true
gap> (-eH[9])*(-eH[10])=eH[9]*eH[10];
```

```
true
gap> ForAll(H,x -> ForAll(H,y -> (-x)*(-y)=x*y));
true
gap> eH[9]*(eH[10]-eH[11])=eH[9]*eH[10]-eH[9]*eH[11];
true
gap> ForAll(H,x -> ForAll(H,y -> ForAll(H,z -> x*(y-z)=x*y-
x*z)));
true
gap> One(H);
Z(13)^0
gap> (-One(H))*eH[12]=-eH[12];
true
gap> ForAll(H,x -> -(One(H))*x=-x);
true
gap> (-One(H))*(-One(H))=One(H);
true
```

Teorem : H birimli halka olsun. $\forall x \in H$ ve her $m,n \in \mathbb{N}$ için x^n elemanı $x^0 = 1$, $x^1 = 1$ ve $x^{n+1} = x^n x$ şeklinde tanımlansın. Bu durumda

$$x^m x^n = x^n x^m = x^{m+n} \text{ ve } (x^m)^n = x^{mn}$$

eşitlikleri sağlanır [3].

Yukarıdaki teoremi GAP programı kullanarak inceleyelim.

```
gap> m:=8;; n:=9;;
gap> eH[3]^m*eH[3]^n=eH[3]^(m+n);
true
gap> (eH[4]^m)^n=eH[4]^(m*n);
true
```

Tanım : H birimli bir halka olsun. $x \in H$ için x nin çarpımsal tersi varsa yani $xy = yx = 1$ olacak şekilde y elemanı varsa x elemanına H nın bir tersiner elemanıdır denir. Bütün tersiner elemanların kümesini $T(H)$ ile gösterilir [3].

GAP programında, bir elemanın bir halka içerisinde ikinci işleme göre tersinin var olup olmadığı `IsUnit()` komutu kullanılarak bulunur. `()` işaretleri sırasıyla bir halka ve bu halka içersinde ikinci işleme göre tersinin olup olmadığı incelenen eleman yazılmalıdır [17].

```
gap> Z6:=ZmodnZ(6);
(Integers mod 6)
gap> IsRing(Z6);
true
gap> eZ6:=Elements(Z6);
[ZmodnZObj(0,6),ZmodnZObj(1,6),ZmodnZObj(2,6),ZmodnZObj(3,6),
ZmodnZObj(4,6),ZmodnZObj(5,6)]
gap> x:=eZ6[4];
ZmodnZObj(3,6)
gap> IsUnit(Z6,x);
false
gap> y:=eZ6[2];
ZmodnZObj(1,6)
gap> IsUnit(Z6,y);
true
```

GAP programında bir halkanın tersinerler kümesini bulmak için `Units()` komutu kullanılır. `()` işaretleri arasına tersiner kümesi bulunmak istenen halka yazılmalıdır.

```
gap> H:=ZmodnZ(6);
(Integers mod 6)
gap> TH:=Units(H);
<group with 1 generators>
gap> F:=ZmodnZ(17);
GF(17)
gap> TF:=Units(F);
<group with 1 generators>
```

Teorem : H birimli bir halka olsun. Bu durumda

$$T(H)=\{x\in H : xy=yx=1\}$$

kümesi H nın ikinci işlemine göre bir gruptur [3].

```
gap> IsGroup(TH);
true
gap> IsGroup(TF);
true
```

Tersinerin en önemli özelliği sadeleşme işlemini gerçekleştirmesidir. Diğer bir deyişle x bir tersiner eleman olmak üzere $xa = xb$ ise $a = b$ dir. Çünkü $x \in T(H)$ ise x in bir $x' = x^{-1}$ ters elemanı vardır ve $x^{-1}xa = x^{-1}xb$ olup $a = b$ elde edilir. Benzer şekilde $ax = bx$ ise $a = b$ olur.

15.1 Alt Halkalar

Tanım : $(H,+,\cdot)$ bir halka olsun. $\varnothing \neq U \subseteq H$ olmak üzere $(U,+,\cdot)$ bir halka ise U ya H nın alt halkası denir ve $U \leq H$ ile gösterilir [3].

Teorem : $(H,+,\cdot)$ bir halka ve $\varnothing \neq U \subseteq H$ olsun. Bu durumda U, H nın bir alt halkası olabilmesi için gerek ve yeter şart

i) $x, y \in U$ ise $x + y \in U$ ve $x \cdot y \in U$,

ii) $x \in U$ ise $-x \in U$

özelliklerinin sağlanmasıdır [3].

GAP programında `Subring()` komutu bir halkanın alt halkasını oluşturmak için kullanılır. `()` işaretleri arasına sırasıyla alt halkası oluşturulacak olan halka ve oluşturulacak alt halka için üreteç yazılmalıdır. `SubringNC()` komutu ile oluşturulacak alt halka için yazılan üreteçlerin halkanın elemanı olup olmadığı test edilmez [17].

```
gap> H:=ZmodnZ(7);
GF(7)
gap> eH:=Elements(H);
[ 0*Z(7), Z(7)^0, Z(7), Z(7)^2, Z(7)^3, Z(7)^4, Z(7)^5 ]
gap> U:=Subring(H,[0*Z(7)]);
<algebra over GF(7)>
gap> IsRing(U);
true
gap> Z(5)^8 in H;
false
```

```
gap> K:=SubringNC(H,[0*Z(5),Z(5)^8]);
GF(5)
gap> IsRing(K);
true
```

Teorem : H bir halka ve $U \leq H$ olsun. Sırasıyla 0_U, 0_H, U ve H nın sıfır elemanları ise $0_U = 0_H$ dır. Ayrıca $x \in U$ ise $(-x)_U = (-x)_H$ dır [3].

Yukarıdaki teoremi GAP programı kullanarak inceleyelim.

```
gap> Zero(U)=Zero(H);
true
gap> One(U)=One(H);
false
```

$(-x)_U = (-x)_H$ özelliğinin sağlanmadığını görmek için bir örnekle daha inceleyelim. Z_{10} halkasını ve bu halkanın bir alt halkası olan $U = \{[0]_{10}, [2]_{10}, [4]_{10}, [6]_{10}, [8]_{10}\}$ alt halkasını alalım. $1_{Z_{10}} = [1]_{10}$ ve $1_U = [6]_{10}$ olduğu ve dolayısıyla $1_{Z_{10}} \neq 1_U$ olduğu açıktır.

```
gap> Z10:=ZmodnZ(10);
(Integers mod 10)
gap> eZ10:=Elements(Z10);
[ ZmodnZObj( 0, 10 ), ZmodnZObj( 1, 10 ), ZmodnZObj( 2, 10 ),
  ZmodnZObj( 3, 10 ), ZmodnZObj( 4, 10 ), ZmodnZObj( 5, 10 ),
  ZmodnZObj( 6, 10 ), ZmodnZObj( 7, 10 ), ZmodnZObj( 8, 10 ),
  ZmodnZObj( 9, 10 ) ]
gap> U:=Subring(Z10,[eZ10[1],eZ10[3],eZ10[5],eZ10[7],eZ10[9]]);
<ring with 5 generators>
gap> eU:=Elements(U);
[ ZmodnZObj( 0, 10 ), ZmodnZObj( 2, 10 ), ZmodnZObj( 4, 10 ),
  ZmodnZObj( 6, 10 ), ZmodnZObj( 8, 10 ) ]
gap> ForAll(U,x -> x*eZ10[7]=x);
true
gap> One(Z10);
ZmodnZObj( 1, 10 )
```

```
gap> One(Z10)=eZ10[2];
true
gap> eZ10[2]=eZ10[7];
false
```

GAP programında bir halkanın diğer bir halkanın alt halkası olup olmadığını incelemek için bir komut yoktur. Aşağıdaki kodlar bir metin editörü içinde yazılabilir ve bu dosya `althalka.gi` adıyla kaydedilebilir. Fonksiyon yazılan ikinci halkanın ilkinin alt halkası olup olmadığını test edecektir.

```
IsSubring := function(arg)
local narg, usage, error, h,u;
narg:= Length(arg);
usage:= "\n Usage: input two propositions; \n";
error:= "\n Error: input positive value; \n";
h:=arg[1];
u:=arg[2];
if ((narg < 2) or (narg >= 3)) then
Print(usage);
return false;
fi;
if ((IsRing(u)=false or IsSubset(h,u)=false) or (u=[])) then
return false;
fi;
if ((IsRing(u)=true and IsSubset(h,u)=true)) then
return true;
fi;
end;
```

Bu dosya `Read` komutu kullanarak GAP programına okutulabilir ve içerisinde ki `IsSubring()` komutu kullanılabilir.

```
gap> Read("althalka.gi");
gap> IsSubring(Z10,U);
true
gap> IsSubring(Rationals,Integers);
```

```
true
```

Tanım : H bir halka ve $X \subseteq H$ olsun.

$$M(X) = \{a \in H : \text{her } x \in X \text{ için } ax = xa\}$$

kümesine X in H halkasındaki merkezi denir [3].

GAP programında `Center()` veya `Centre()` komutu bir halkanın merkezini bulmak için kullanılır [17].

```
gap> H:=Ring(Z(7));
GF(7)
gap> C:=Center(H);
GF(7)
gap> IsRing(C);
true
```

Örnek : $M(X)$ kümesi H nın bir alt halkasıdır [3].

```
gap> IsSubring(H,C);
true
```

Tanım : H bir halka olsun. $x \in H$ olmak üzere $x^2 = x$ özelliğini sağlayan elemenlara idempotent eleman denir [10].

GAP programında `Idempotents()` komutu bir halkanın idempotent elemanlarını bulmak için kullanılır [17].

```
gap> H:=ZmodnZ(14);
(Integers mod 14)
gap> Idempotents(H);
[ ZmodnZObj( 0, 14 ), ZmodnZObj( 1, 14 ), ZmodnZObj( 7, 14 ),
  ZmodnZObj( 8, 14 ) ]
gap> Size(Idempotents(H));
4
```

Tanım : H bir halka olsun. $x \in H$ olmak üzere bazı m pozitif tamsayılar için $x^m = 0$ özelliğini sağlayan elemenlara nilpotent eleman denir [10].

GAP programında bir halkanın nilpotent elemanlarını bulabilmek için bir fonksiyon yazılabilir. Aşağıdaki kodlar bir metin editörü içinde yazılabilir ve bu dosya `nilpotent.gi` adıyla kaydedilebilir.

```
Nilpotent:= function(R)
local a,x;
a:=Size(R);
return Filtered(R,x -> IsZero(x^a));
end;
```

Bu dosya `Read` komutu kullanarak GAP programına okutulabilir ve içerisinde ki `Nilpotent()` komutu kullanılabilir.

```
gap> Read("nilpotent.gi");
gap> Nilpotent(U);
[ 0*Z(3) ]
gap> Nilpotent(H);
[ ZmodnZObj( 0, 14 ) ]
gap> Nilpotent(N);
[ ZmodnZObj( 0, 9 ), ZmodnZObj( 3, 9 ), ZmodnZObj( 6, 9 ) ]
gap> Size(Nilpotent(N));
3
```

15.2 İdeal

Tanım : H bir halka ve $A \subseteq H$ olsun. Eğer

i) $(A,+) \leq (H,+)$ (yani $x,y \in A$ ise $x-y \in A$)

ii) $a \in A$ ve $x \in H$ ise $ax \in A$ ve $xa \in A$

özellikleri sağlanıyorsa A ya H nın bir ideali denir ve $A \lhd H$ ile gösterilir [3].

GAP programında `Ideal()` komutu bir halkanın idealini oluşturmak için kullanılır. `()` işaretleri arasına sırasıyla ideali oluşturulacak olan halka ve oluşturulacak ideal için üreteç yazılmalıdır. `IdealNC()` versiyonunda oluşturulacak ideal için yazılan üreteçlerin halkanın elemanı olup olmadığı test edilmez. GAP programında `IsIdeal()` komutu verilen bir halkanın diğerinin ideali olup olmadığını hesaplamak için kullanılır [17].

```
gap> H:=Integers;
Integers
gap> I:=Ideal(H,[2]);
<two-sided ideal in Integers, (1 generators)>
gap> K:=Ring(Z(8));
GF(2^3)
gap> eK:=Elements(K);
[0*Z(2),Z(2)^0,Z(2^3),Z(2^3)^2,Z(2^3)^3,Z(2^3)^4,Z(2^3)^5,
Z(2^3)^6 ]
gap> I:=Ideal(K,[Z(2^3)^2,Z(2^3)^3]);
<two-sided ideal in GF(2^3), (2 generators)>
gap> IsIdeal(K,I);
true
gap> Z(4) in eK;
false
gap> J:=IdealNC(K,[Z(4)]);
<two-sided ideal in GF(2^3), (1 generators)>
```

Uyarı : H bir halka olmak üzere aşağıdaki özellikler sağlanır.

i) H nın her zaman $\{0\}$ ve kendisi olmak üzere en az iki ideali vardır.

ii) Her ideal bir alt halkadır. Çünkü her ideal çarpımsal ikili işlem özelliğine sahiptir. Fakat tersi doğru değildir. Örneğin $\mathbb{Z}$, $\mathbb{Q}$ nun bir alt halkası olmasına rağmen bir ideal değildir. Çünkü $1/2 \in \mathbb{Q}$ ve $1 \in \mathbb{Z}$ ise $(1/2)\cdot 1 = 1/2 \notin \mathbb{Z}$ dir.

iii) H birimli bir halka ve $1_H \in A$ ise $A = H$ dır. Gerçekten de, herhangi $h \in H$ için A ideal olduğundan $h = h\cdot 1 \in A$ olup $H \subseteq A$ olur. O halde $A = H$ olur.

iv) A ideali boş küme olamaz. Çünkü $(A,+)$ altgrup olduğundan $0_H \in A$ olmak zorundadır [3].

Örnek : Herhengi $n \in \mathbb{Z}$ için

$$n\mathbb{Z} = \{nk : k \in \mathbb{Z}\}$$

kümesi $\mathbb{Z}$ nin bir idealidir.

i) $x, y \in n\mathbb{Z}$ için $x - y = nk - nk' = n(k - k') = nk_1 \in n\mathbb{Z}$;

ii) $a \in n\mathbb{Z}$ ve $x \in \mathbb{Z}$ için $xa = x(nk) = n(xk) \in n\mathbb{Z}$ ve $\mathbb{Z}$ abelyen olduğundan $ax \in n\mathbb{Z}$ dir. Bundan başka $\mathbb{Z}$ nin her ideali $n\mathbb{Z}$ formdadır [3].

16 TAMLIK BÖLGESİ VE CİSİMLER

Tanım : H bir halka olsun. H halkasında $xy = 0$ veya $yx = 0$ olacak şekilde $y \neq 0$ elemanı varsa $x \neq 0 \in H$ elemanına H nın bir sıfır böleni denir [3].

Tanım : Sıfır bölen içermeyen halkalara tam halka denir [7].

GAP programında bir halkanın sıfır bölen içerip içermediği yada tam halka olup olmadığı `IsIntegralRing()` komutu kullanılarak bulunur. Bu komut, `true` değerini verirse halkanın sıfır bölen içermediği (yani tam halka) olduğunu anlaşılır. Aksi durumda komut `false` değerini verirse , halka sıfır bölen içerir. (yani tam halka değildir) [17]

```
gap> H:=Ring(Z(2));
GF(2)
gap> IsIntegralRing(H);
true
gap> U:=ZmodnZ(6);
(Integers mod 6)
gap> IsIntegralRing(U);
false
gap> eU:=Elements(U);
[ ZmodnZObj( 0, 6 ), ZmodnZObj( 1, 6 ), ZmodnZObj( 2, 6 ),
ZmodnZObj( 3, 6 ), ZmodnZObj( 4, 6 ), ZmodnZObj( 5, 6 ) ]
gap> eU[4]=Zero(U);
false
gap> eU[5]=Zero(U);
false
gap> eU[4]*eU[5]=Zero(U);
true
gap> IsIntegralRing(Integer
true
```

Teorem : H bir halka olsun. $\forall x, y, a \in H$ olmak üzere H nın sıfır bölenlerinin olmaması için gerek ve yeter koşul

$$ax = ay,\ a \neq 0 \Rightarrow x = y,$$
$$xa = ya,\ a \neq 0 \Rightarrow x = y$$

özelliklerinin sağlanmasıdır [3].

GAP programında bir halkasının sıfır bölenlerini bulmak için bir program yazılabilir. Aşağıdaki kodlar bir metin editörü içinde yazılabilir ve bu dosya `bolen.gi` adıyla kaydedilebilir. Fonksiyon verilen bir halkanın sıfır bölenlerini hesaplayacak olup, halkanın sıfır bölen içermemesi durumunda bunun bir tam halka olduğu ekrana yazılacaktır.

```
SBolen := function(h)
local x, y, a, sifir, dikkat, hata ;
sifir:=[];
hata:= " Hata : Cebirsel yapi bir halka olmalidir. \n";
dikkat:= " Dikkat : Verilen halka bir tam halka oldugundan sifir
bolen icermez. \n";
if (IsRing(h)<>true) then
Print(hata);
return false;
fi;
if (IsIntegralRing(h)) then
Print(dikkat);
return false;
fi;
a:=Filtered(h,x->x<>Zero(h));
for x in a do
for y in a do
if x*y=Zero(h) then
Append(sifir,[x,y]);
fi;
od;
od;
return sifir;
end;
```

Bu dosya `Read` komutu kullanarak GAP programına okutulabilir ve içerisinde ki `SBolen()` komutu kullanılabilir.

```
gap> Read("bolen.gi");
```

```
gap> SBolen(Integers);
 Dikkat : Verilen halka bir tam halka oldugundan sifir bolen
icermez.
false
gap> SBolen(ZmodnZ(4));
[ZmodnZObj( 2, 4 ),ZmodnZObj( 2, 4 )]
gap> SBolen(ZmodnZ(8));
[ZmodnZObj( 2, 8 ),ZmodnZObj( 4, 8 ),ZmodnZObj( 4, 8 ),
ZmodnZObj( 2, 8 ),ZmodnZObj( 4, 8 ),ZmodnZObj( 4, 8 ),
ZmodnZObj( 4, 8 ), ZmodnZObj( 6, 8 ),ZmodnZObj( 6, 8 ),
ZmodnZObj( 4, 8 ) ]
```

16.1 Bir Halkanın Karakteristiği

Tanım : H birimli bir halka olsun. $n \cdot 1 = 0$ olacak şekilde en küçük $n \in \mathbb{N}$ elemanına (varsa) H halkasının karakteristiği denir ve $Kar(H) = n$ biçiminde gösterilir. Böyle bir doğal sayı yoksa H nın karakteristiği sıfır olarak tanımlanır [3].

GAP programında `Characteristic()` komutu verilen bir birimli halkanın karakterisitiğini bulmak için kullanılır [17].

```
gap> H:=Ring(Z(7));
GF(7)
gap> Characteristic(H);
7
gap> 7*One(H)=Zero(H);
true
gap> U:=ZmodnZ(6);
(Integers mod 6)
gap> Characteristic(U);
6
gap> 6*One(U)=Zero(U);
true
gap> Characteristic(Rationals);
0
```

Teorem : H birimli bir halka ve $Kar(H) = n$ olsun. Bu durumda H nın sıfır böleni yok ise n sıfır veya bir asal sayıdır [3].

Yukarıdaki teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=Ring(Z(173));
GF(173)
gap> IsIntegralRing(H);
true
gap> n:=Characteristic(H);
173
gap> IsPrime(n);
true
gap> IsIntegralRing(Integers);
true
gap> n:=Characteristic(Integers);
0
```

Tanım : H birimli bir halka olsun. Eğer

i) H nın bir sıfır böleni yok,

ii) H değişmeli,

ise H ya bir tamlık bölgesi denir [3].

Diğer bir deyişle birimli ve değişmeli tam halkaya tamlık bölgesi denir.

16.2 Cisim

Tanım : Birimli değişmeli bir H halkasının sıfırdan farkı her elemanı tersiner ise H ya bir cisim denir [3].

GAP programında `Field()` ve `DefaultField()` komutları cisim yapısı oluşturmak için kullanılır. `()` işaretleri arasına bu cisim içerisinde yer alacak elemanlardan bir veya birkaçı sırayla veya liste halinde yazılmalıdır [17].

```
gap> Field(1,-1);
Rationals
gap> DefaultField(1,-1);
Rationals
```

```
gap> Field([1..5]);
Rationals
gap> DefaultField([1..5]);
Rationals
```

`Field()` ve `DefaultField()` komutları arasındaki temel fark `Field()` komutunda verilen elemanları içeren en küçük cismin bulunabilmesi `DefaultField()` komutunda ise verilen elemanları içeren daha büyük cismin bulunabilmesidir. GAP programında `IsField()` komutu, cebirsel yapının cisim olup olmadığını test etmek için kullanılır [17].

```
gap> F:=Field(Z(7));
GF(7)
gap> IsField(F);
true
gap> H:=ZmodnZ(6);
(Integers mod 6)
gap> IsField(H);
false
```

Teorem : Her cisim bir tamlık bölgesi her sonlu tamlık bölgesi bir cisimdir [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> F:=Rationals;
Rationals
gap> IsField(F);
true
gap> IsIntegralRing(F);
true
gap> T:=ZmodnZ(7);
GF(7)
gap> IsIntegralRing(T);
true
gap> Size(T);
7
```

```
gap> IsFinite(T);
true
gap> IsField(T);
true
gap> U:=Integers;
Integers
gap> IsIntegralRing(U);
true
gap> IsFinite(U);
false
gap> IsField(U);
false
```

16.3 Alt Cisimler

Tanım : F bir cisim ve $\varnothing \neq E \subseteq F$ olsun. F nin işlemlerine göre E bir cisim ise E ve F nin alt cismi denir [3].

GAP programında `Subfield()` komutu bir cismin alt cismini oluşturmak için kullanılır. `()` işaretleri arasına sırasıyla alt cismi oluşturulacak olan cisim ve oluşturulacak alt cisim için üreteç yazılmalıdır. `SubfieldNC()` versiyonu oluşturulacak alt cisim için yazılan üreteçlerin cismin elemanı olup olmadığı test edilmez [17].

```
gap> F:=Field(Z(17));
GF(17)
gap> eF:=Elements(F);
[0*Z(17),Z(17)^0,Z(17),Z(17)^2,Z(17)^3,Z(17)^4,Z(17)^5,Z(17)^6,
Z(17)^7,Z(17)^8,Z(17)^9,Z(17)^10,Z(17)^11,Z(17)^12,Z(17)^13,
Z(17)^14, Z(17)^15 ]
gap> K:=Subfield(F,[eF[10]]);
GF(17)
```

Teorem : F bir cisim ve $\varnothing \neq E \subseteq F$ olsun. E, F nin alt cismi olabilmesi için gerek ve yeter koşul

i) $x, y \in E$ ise $x+y \in E$ ve $xy \in E$

ii) $x \in E$ ise $-x \in E$ ve $x \neq 0$ için $x^{-1} \in E$

olmasıdır [3].

GAP programında bir cismin diğer bir cismin alt cisimi olup olmadığını incelemek için bir komut yoktur. Aşağıdaki kodlar bir metin editörü içinde yazılabilir ve bu dosya `altcisim.gi` adıyla kaydedilebilir. Fonksiyon yazılan ikinci cismin ilkinin alt cisimi olup olmadığını test edecektir.

```
IsSubfield:= function(arg)
local narg, usage, error, h,u;
narg:= Length(arg);
usage:= "\n Usage: input two propositions; \n";
error:= "\n Error: input positive value; \n";
h:=arg[1];
u:=arg[2];
if ((narg < 2) or (narg >= 3)) then
Print(usage);
return false;
fi;
if ((IsField(u)=false or IsSubset(h,u)=false) or (u=[])) then
return false;
fi;
if ((IsField(u)=true and IsSubset(h,u)=true)) then
return true;
fi;
end;
```

Bu dosya `Read` komutu kullanarak GAP programına okutulabilir ve içerisinde ki `IsSubfield()` komutu kullanılabilir.

```
gap> Read("altcisim.gi");
gap> IsSubfield(F,K);
true
gap> IsSubfield(Rationals,Integers);
false
```

Uyarı : E, F nin bir alt cismi ise $1_E = 1_F$ dir. Her $x \in E$ için $x1_E = x1_F$ dir. Fakat (E sıfırdan farklı en az bir eleman içerdiğinden) $x \neq 0_F$ için $x_F^{-1} \in F$ dir.

$$x_F^{-1}x1_E = x_F^{-1}x1_F \Rightarrow 1_F1_E = 1_E1_F$$

olup $1_E = 1_F$ elde edilir. Benzer şekilde her $0 \neq x \in E$ için $(x^{-1})_E = (x^{-1})_F$ dir [3].

```
gap> One(F)=One(K);
true
```

Örnek : $\mathbb{Z}_{[i]} = \{a+ib \mid a,b \in \mathbb{Z}\}$ kümesi kompleks sayılardaki toplama ve çarpma işlemlerine göre bir halkadır ve tamlık bölgesidir [3]. Bu halkaya Gaussian halkası denir.

GAP programında `GaussianIntegers` komutu Gaussian halkalarını tanımlamak için kullanılırlar. Gaussian halkasında bir tamlık bölgesi olmasına karşın bir cisim değildir. Çünkü bu halkada sıfırdan farklı her elemanın tersi yoktur. Kompleks sayılarda kullanılan i sayısı GAP programında `E(4)` olarak tanımlanır [17].

```
gap> i:=E(4);
E(4)
gap> i^2;
-1
gap> H:=GaussianIntegers;
GaussianIntegers
gap> IsRing(H);
true
gap> IsIntegralRing(H);
true
gap> Units(H);
[ -1, 1, -E(4), E(4) ]
gap> IsField(H);
false
```

Tanım : F bir cisim olsun. F cisminin kendisinden başka alt cismi yok ise F ye asal cisim denir [13].

GAP programında `IsPrimeField()` komutu bir cismin asal cisim olup olmadığını test eder. `()` işaretleri arasına asal cisim olup olmadığı test edilecek olan cisim yazılmalıdır [17].

```
gap> IsPrimeField(Rationals);
true
gap> F:=Field(Z(7));
GF(7)
gap> IsPrimeField(F);
true
gap> T:=Field(Z(4));
GF(2^2)
gap> IsPrimeField(T);
false
```

Teorem : F bir cisim olsun. F nin tek bir K asal alt cismi vardır [13].

GAP programında `PrimeField()` komutu bir cismin asal alt cismini bulmak için kullanılır. `()` işaretleri arasına asal alt cismi bulunacak olan cisim yazılmalıdır [17].

```
gap> K:=PrimeField(F);
GF(7)
gap> IsSubfield(F,K);
true
gap> U:=PrimeField(T);
GF(2)
gap> IsSubfield(T,U);
true
gap> M:=Field(Z(9));
GF(3^2)
gap> N:=PrimeField(M);
GF(3)
gap> IsSubfield(M,N);
true
```

17 HALKA HOMOMORFİZMLERİ

Tanım : H ve K iki halka ve $f : H \to K$ bir fonksiyon olsun. Her $x, y \in H$ için

i) $f(x+y) = f(x) + f(y)$

ii) $f(xy) = f(x)f(y)$

ise f ye H dan K ya bir halka homomorfizmi denir [3].

GAP programında fonksiyon oluşturmak için `MappingByFunction()` komutunun kullanılabileceğinden bahsedilmişti. Bir halka homomorfizmi oluşturmak için görüntü ve değer kümelerinin yerine birer halka alınarak bir fonksiyon oluşturulur. Oluşturulan fonksiyonun halka homomorfizmi olup olmadığı `IsRingHomomorphism()` komutu kullanılarak test edilir. `IsRingHomomorphism()` komutu yalnız sonlu elemanlı halkalar için kulanılabilir [17].

Örnek : $\forall x \in H$ için $f(x) = 0_K$ olarak tanımlı $f = H \to K$ fonksiyonu bir homomorfizmadır. Özel olarak bu homomorfizmaya sıfır homomorfizmi denir [3].

GAP programı kullanılarak sıfır homomorfizma tanımlanabilir.

```
gap> H:=Ring(Z(4));
GF(2^2)
gap> K:=Ring(Z(8));
GF(2^3)
gap> h:=x->Zero(K);
function( x ) ... end
gap> f:=MappingByFunction(H,K,h);
MappingByFunction( GF(2^2), GF(2^3), function( x ) ... end )
gap> IsRingHomomorphism(f);
true
```

Örnek : n herhangi bir pozitif tamsayı olmak üzere $x \to x \bmod n$ biçiminde tanımlanacak olan fonksiyon $\mathbb{Z}$ den $\mathbb{Z}_n$ e bir halka homomorfizmidir.

GAP programı kullanılarak doğal homomorfizma tanımlanabilir. $\mathbb{Z}$ halkası sonlu elemanlı olmadığından `IsRingHomomorphism` komutu kullanılarak oluşturulan fonksiyonun halka homomorfizmi olup olmadığı sorulursa GAP programı hata verecektir.

```
gap> H:=Integers;
Integers
gap> K:=Ring(Z(5));
GF(5)
gap> h:=x->x mod 5;
function( x ) ... end
gap> f:=MappingByFunction(H,K,h);
MappingByFunction( Integers, GF(5), function( x ) ... end )
gap> IsRingHomomorphism(f);
Error, no method found! For debugging hints type ?Recovery from
NoMethodFound
Error, no 2nd choice method found for `RespectsMultiplication'
on 1 arguments \
called from
<function>( <arguments> ) called from read-eval-loop
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

Örnek : H birimli bir halka olsun.

$$f : \mathbb{Z} \to H$$
$$n \mapsto n.1$$

fonksiyonu bir homomorfizmdir [3].

GAP programı kullanılarak bu homomorfizma tanımlanabilir. $\mathbb{Z}$ halkası sonlu elemanlı olmadığından `IsRingHomomorphism` komutu kullanılarak oluşturulan fonksiyonun halka homomorfizmi olup olmadığı sorulursa GAP programı hata verecektir.

```
gap> H:=Integers;
Integers
gap> K:=Ring(Z(13));
```

```
GF(13)
gap> One(K);
Z(13)^0
gap> h:=x->x*One(K);
function( x ) ... end
```

Örnek : $\mathbb{Z}_{10}$ halkasından $\mathbb{Z}_{10}$ halkasına $f(x)=5x$ biçiminde tanımlanan fonksiyon bir halka homomorfizmasıdır. $f(x)=2x$ fonksiyonu ise bir halka homomorfizması değildir. Buradaki 2 ve 5 sayıları mod10 a göre düşünülmelidir [9].

GAP programı kullanılarak bu fonksiyonlar tanımlanabilir.

```
gap> H:=ZmodnZ(10);
(Integers mod 10)
gap> eH:=Elements(H);
[ ZmodnZObj( 0, 10 ), ZmodnZObj( 1, 10 ), ZmodnZObj( 2, 10 ),
  ZmodnZObj( 3, 10 ), ZmodnZObj( 4, 10 ), ZmodnZObj( 5, 10 ),
  ZmodnZObj( 6, 10 ), ZmodnZObj( 7, 10 ), ZmodnZObj( 8, 10 ),
  ZmodnZObj( 9, 10 ) ]
gap> h:=x->eH10[6]*x;
function( x ) ... end
gap> f:=MappingByFunction(H,H,h);
MappingByFunction( (Integers mod 10), (Integers mod
10), function( x ) ... end )
gap> IsRingHomomorphism(f);
true
gap> k:=x->eH[3]*x;
function( x ) ... end
gap> g:=MappingByFunction(H,H,k);
MappingByFunction( (Integers mod 10), (Integers mod
10), function( x ) ... end )
gap> IsRingHomomorphism(g);
false
```

Teorem : $f : H \to K$ bir homomorfizma ise $f(0_H) = 0_K$ dır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=Ring(Z(4));
GF(2^2)
gap> K:=Ring(Z(8));
GF(2^3)
gap> h:=x->x^2;
function( x ) ... end
gap> f:=MappingByFunction(H,K,h);
MappingByFunction( GF(2^2), GF(2^3), function( x ) ... end )
gap> IsRingHomomorphism(f);
true
gap> Image(f,Zero(H))=Zero(K)
true
```

Teorem : $f : H \to K$ bir homomorfizma ise $f(-a) = f(a)$ dır [7].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> ForAll(H,x->Image(f,-x)=Image(f,x));
true
```

Teorem : H ve K birimli halkalar olsalar bile $f(1_H) = f(1_K)$ doğru olmayabilir. f fonksiyonu örten ise $f(1_H) = f(1_K)$ özelliği sağlanır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=Ring(Z(4));
GF(2^2)
gap> K:=Ring(Z(2));
GF(2)
gap> h:=x->x;
function( x ) ... end
gap> f:=MappingByFunction(H,K,h);
MappingByFunction( GF(2^2), GF(2), function( x ) ... end )
```

```
gap> IsSurjective(f);
true
gap> One(H)=One(K);
true
gap> IsAbelian(H);
true
gap> IsAbelian(K);
true
```

Teorem : $f: H \to K$ bir halka homomorfizmi olsun. Bu durumda U, H nın bir alt halkası ise $f(U) = \{f(x) : x \in U\}$ kümesi K nın bir alt halkasıdır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> Z10:=ZmodnZ(10);
(Integers mod 10)
gap> eZ10:=Elements(Z10);
[ ZmodnZObj( 0, 10 ), ZmodnZObj( 1, 10 ), ZmodnZObj( 2, 10 ),
  ZmodnZObj( 3, 10 ), ZmodnZObj( 4, 10 ), ZmodnZObj( 5, 10 ),
  ZmodnZObj( 6, 10 ), ZmodnZObj( 7, 10 ), ZmodnZObj( 8, 10 ),
  ZmodnZObj( 9, 10 ) ]
gap> U:=Subring(Z10,[eZ10[1],eZ10[3],eZ10[5],eZ10[7],eZ10[9]]);
<ring with 5 generators>
gap> Read("althalka.gi");
gap> K:=ZmodnZ(2);
GF(2)
gap> k:=x->Zero(K);
function( x ) ... end
gap> h:=MappingByFunction(Z10,K,k);
MappingByFunction( (Integers mod 10), GF(2), function( x ) ...
end )
gap> IsRingHomomorphism(h);
true
gap> Image(h,U);
[ 0*Z(2) ]
gap> hU:=Ring(Image(h,U));
```

```
<algebra over GF(2)>
gap> IsRing(hU);
true
gap> IsSubring(K,hU);
true
```

Teorem : $f : H \to K$ bir halka homomorfizmi olsun. Bu durumda

$$Çekf = \{x \in H : f(x) = 0_K\}$$

kümesi H nın bir alt halkasıdır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=Ring(Z(4));
GF(2^2)
gap> K:=Ring(Z(8));
GF(2^3)
gap> h:=x->x^2;
function( x ) ... end
gap> f:=MappingByFunction(H,K,h);
MappingByFunction( GF(2^2), GF(2^3), function( x ) ... end )
gap> IsRingHomomorphism(f);
true
gap> C:=Ring(Elements(Kernel(f)));
<algebra of dimension 0 over GF(2)>
gap> IsSubring(H,C);
true
```

Teorem : $f : H \to K$ bir halka homomorfizmi olsun. V , K nın bir alt halkası ise

$$f^{-1}(V) = \{x \in H : f(x) \in V\}$$

kümesi H nın bir alt halkasıdır [3].

Bu teoremi GAP programı kullanarak inceleyelim.

```
gap> H:=ZmodnZ(10);
(Integers mod 10)
```

```
gap> f:=MappingByFunction(H,H,h);
MappingByFunction( (Integers mod 10), (Integers mod
10), function( x ) ... end )
gap> h:=x->x;
function( x ) ... end
gap> f1:=InverseGeneralMapping(f);
InverseGeneralMapping( MappingByFunction( (Integers mod 10),
(Integers mod 10), function( x ) ... end ) )
gap> eZ10:=Elements(H);
[ ZmodnZObj( 0, 10 ), ZmodnZObj( 1, 10 ), ZmodnZObj( 2, 10 ),
  ZmodnZObj( 3, 10 ), ZmodnZObj( 4, 10 ), ZmodnZObj( 5, 10 ),
  ZmodnZObj( 6, 10 ), ZmodnZObj( 7, 10 ), ZmodnZObj( 8, 10 ),
  ZmodnZObj( 9, 10 ) ]
gap> V:=Subring(H,[eZ10[1],eZ10[3],eZ10[5],eZ10[7],eZ10[9]]);
<ring with 5 generators>
gap> HV:=Ring(Image(f1,V));
<ring with 5 generators>
gap> IsSubring(H,HV);
true
```

18 LATİSLER

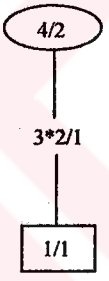
18.1 Giriş

Bu bölümde, bu çalışmayı inceleyen okuyucuların GAP programı açısından yeni gruplar üretip işlemler gerçekleştirebilmesi için , 30. Dereceye kadar olan grupların GAP programı sıralamalarına göre üreteçleri listeler halinde verilmiştir.

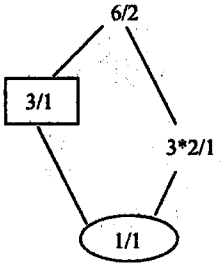
Sıralama da GAP programı sıralaması izlenmiş olup her grup için latisler ayrı ayrı çizilmiştir. Tablolarda iki tane grup adı yer almakta olup birincisi [18] de belirtilen kitaptan alınmış olup, ikincisi ise GAP programı ismi olarak yer almıştır. Tablolarda üreteçlerin yanı sıra her grubun abelyen olup olmaması, grubun derecesi ve grubun elemanları da listeler halinde sunulmuştur.

18.2 Grup Latisleri

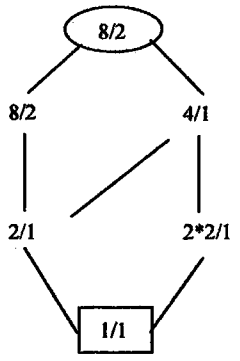
Grubun Tipi	4/2
Grubun Adı	C_2xC_2 - Elementary
GAP Tipi	4/1
GAP Adı	k4
Derecesi	4
Abelyenlik	Abelyen
Üreteci	(1,2),(3,4)
Elemanları	[(),(3,4),(1,2),(1,2)(3,4)]



Grubun Tipi	6/2
Grubun Adı	S_3 - Simetrik
GAP Tipi	6/2
GAP Adı	s3
Derecesi	6
Abelyenlik	Abelyen Değil
Üreteci	(1,2),(2,3)
Elemanları	[(),(2,3),(1,2),(1,2,3), (1,3,2),(1,3)]

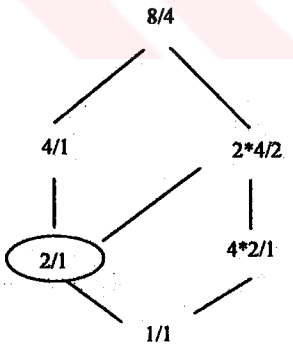


Grubun Tipi	8/2
Grubun Adı	C_4xC_2
GAP Tipi	8/2
GAP Adı	c4c2
Derecesi	8
Abelyenlik	Abelyen
Üreteci Elemanları	(1,2,3,4),(5,6)
	[(),(5,6),(1,2,3,4), (1,2,3,4)(5,6),(1,3)(2,4), (1,3)(2,4)(5,6),(1,4,3,2), (1,4,3,2)(5,6)]

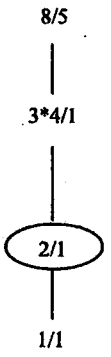


Grubun Tipi	8/3
Grubun Adı	$C_2xC_2xC_2$ - Elementary
GAP Tipi	8/1
GAP Adı	$c2^3$
Derecesi	8
Abelyenlik	Abelyen
Üreteci Elemanları	(1,2),(3,4),(5,6)
	[(),(5,6),(3,4),(3,4)(5,6), (1,2),(1,2)(5,6),(1,2)(3,4), (1,2)(3,4)(5,6)]

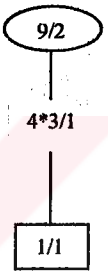
Grubun Tipi	8/4
Grubun Adı	D_4 - Dihedral
GAP Tipi	8/4
GAP Adı	d8
Derecesi	8
Abelyenlik	Abelyen Değil
Üreteci Elemanları	(1,2,3,4),(1,3)
	[(),(2,4),(1,2)(3,4),(1,2,3,4), (1,3),(1,3)(2,4),(1,4,3,2), (1,4)(2,3)]



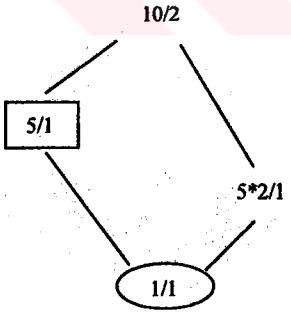
Grubun Tipi	8/5
Grubun Adı	Q - Quaternion
GAP Tipi	8/5
GAP Adı	q8
Derecesi	8
Abelyenlik	Abelyen Değil
Üreteci	(1,5,2,6)(3,7,4,8),(1,7,2,8)(3,6,4,5)
Elemanları	[(),(1,2)(3,4)(5,6)(7,8),(1,3,2,4)(5,8,6,7),(1,4,2,3)(5,7,6,8),(1,5,2,6)(3,7,4,8),(1,6,2,5)(3,8,4,7),(1,7,2,8)(3,6,4,5),(1,8,2,7)(3,5,4,6)]



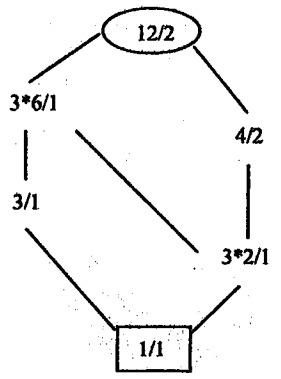
Grubun Tipi	9/2
Grubun Adı	C_3xC_3 - Elementary
GAP Tipi	9/1
GAP Adı	$c3^2$
Derecesi	9
Abelyenlik	Abelyen
Üreteci	(1,2,3),(4,5,6)
Elemanları	[(),(4,5,6),(4,6,5),(1,2,3),(1,2,3)(4,5,6),(1,2,3)(4,6,5),(1,3,2),(1,3,2)(4,5,6),(1,3,2)(4,6,5)]



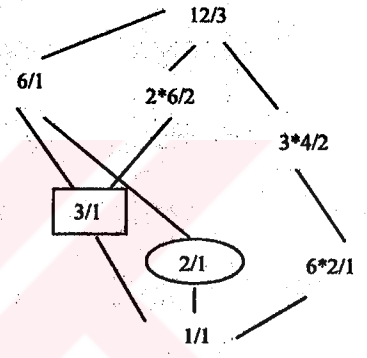
Grubun Tipi	10/2
Grubun Adı	D_5 - Dihedral
GAP Tipi	10/2
GAP Adı	d10
Derecesi	10
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5),(2,5)(3,4)
Elemanları	[(),(2,5)(3,4),(1,2)(3,5),(1,2,3,4,5),(1,3)(4,5),(1,3,5,2,4),(1,4)(2,3),(1,4,2,5,3),(1,5,4,3,2),(1,5)(2,4)]



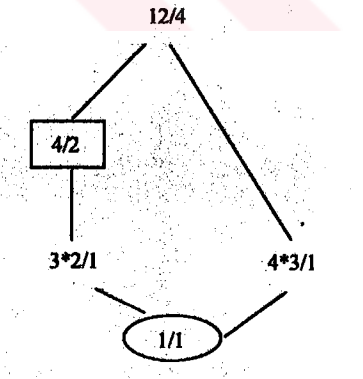
Grubun Tipi	12/2
Grubun Adı	C_6xC_2
GAP Tipi	12/1
GAP Adı	c6c2
Derecesi	12
Abelyenlik	Abelyen
Üreteci	(1,2,3),(4,5),(6,7)
Elemanları	[(),(6,7),(4,5),(4,5)(6,7), (1,2,3),(1,2,3)(6,7),(1,2,3) (4,5),(1,2,3)(4,5)(6,7), (1,3,2),(1,3,2)(6,7),(1,3,2) (4,5),(1,3,2)(4,5)(6,7)]



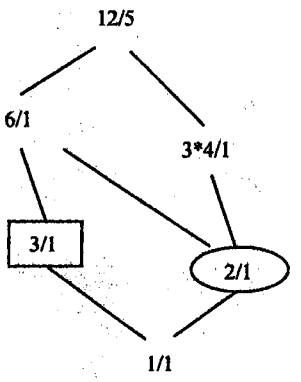
Grubun Tipi	12/3
Grubun Adı	D_6 - Dihedral
GAP Tipi	12/3
GAP Adı	d12
Derecesi	12
Abelyenlik	Abelyen Değil
Üreteci	(1,2),(2,3),(4,5)
Elemanları	[(),(4,5),(2,3),(2,3)(4,5), (1,2),(1,2)(4,5),(1,2,3), (1,2,3)(4,5),(1,3,2),(1,3,2) (4,5),(1,3),(1,3)(4,5)]



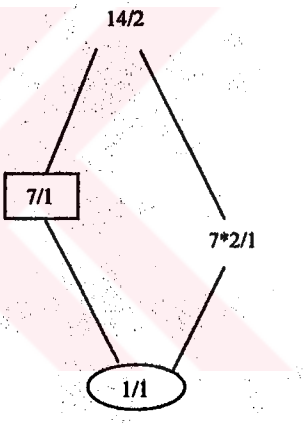
Grubun Tipi	12/4
Grubun Adı	A_4 - Alterne
GAP Tipi	12/5
GAP Adı	a4
Derecesi	12
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3),(4,5),(6,7)
Elemanları	[(),(2,3,4),(2,4,3),(1,2) (3,4),(1,2,3),(1,2,4),(1,3,2), (1,3,4),(1,3)(2,4),(1,4,2), (1,4,3),(1,4)(2,3)]



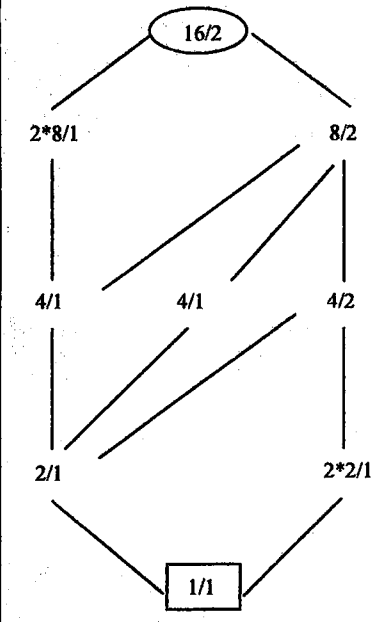
Grubun Tipi	12/5
Grubun Adı	Q_6 - Dicyclic
GAP Tipi	12/4
GAP Adı	q12
Derecesi	12
Abelyenlik	Abelyen Değil
Üreteci	(1,2)(3,4)(5,6,7),(1,3,2,4)(6,7)
Elemanları	[(),(5,6,7),(5,7,6),(1,2)(3,4),(1,2)(3,4)(5,6,7),(1,2)(3,4)(5,7,6),(1,3,2,4)(6,7),(1,3,2,4)(5,6),(1,3,2,4)(5,7),(1,4,2,3)(6,7),(1,4,2,3)(5,6),(1,4,2,3)(5,7)]



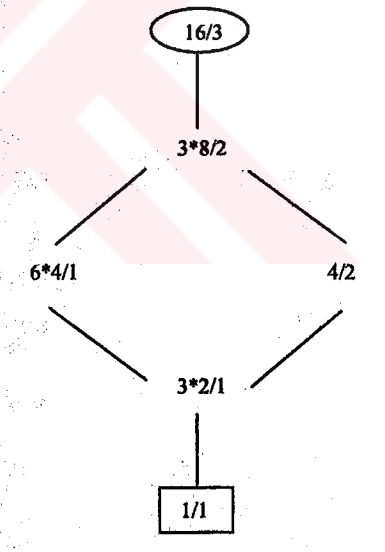
Grubun Tipi	14/2
Grubun Adı	D_7 - Dihedral
GAP Tipi	14/2
GAP Adı	d14
Derecesi	14
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7),(2,7)(3,6)(4,5)
Elemanları	[(),(2,7)(3,6)(4,5),(1,2)(3,7)(4,6),(1,2,3,4,5,6,7),(1,3)(4,7)(5,6),(1,3,5,7,2,4,6),(1,4)(2,3)(5,7),(1,4,7,3,6,2,5),(1,5)(2,4)(6,7),(1,5,2,6,3,7,4),(1,6)(2,5)(3,4),(1,6,4,2,7,5,3),(1,7,6,5,4,3,2),(1,7)(2,6)(3,5)]



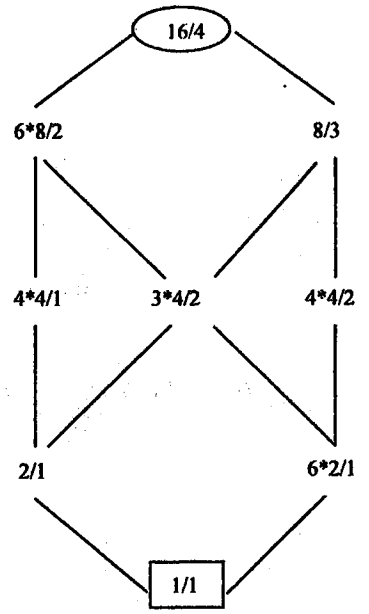
Grubun Tipi	16/2
Grubun Adı	C_2xC_8
GAP Tipi	16/3
GAP Adı	c8c2
Derecesi	16
Abelyenlik	Abelyen
Üreteci	(1,2),(3,4,5,6,7,8,9,10)
Elemanları	[(),(3,4,5,6,7,8,9,10), (3,5,7,9)(4,6,8,10), (3,6,9,4,7,10,5,8),(3,7)(4,8) (5,9)(6,10),(3,8,5,10,7,4,9,6) ,(3,9,7,5)(4,10,8,6), (3,10,9,8,7,6,5,4),(1,2),(1,2) (3,4,5,6,7,8,9,10),(1,2) (3,5,7,9)(4,6,8,10),(1,2) (3,6,9,4,7,10,5,8),(1,2)(3,7) (4,8)(5,9)(6,10),(1,2) (3,8,5,10,7,4,9,6),(1,2) (3,9,7,5)(4,10,8,6),(1,2) (3,10,9,8,7,6,5,4)]



Grubun Tipi	16/3
Grubun Adı	C_4xC_4 - Homocyclic
GAP Tipi	16/4
GAP Adı	$c4^2$
Derecesi	16
Abelyenlik	Abelyen
Üreteci	(1,2,3,4),(5,6,7,8)
Elemanları	[(),(5,6,7,8),(5,7)(6,8), (5,8,7,6),(1,2,3,4), (1,2,3,4)(5,6,7,8),(1,2,3,4) (5,7)(6,8),(1,2,3,4)(5,8,7,6), (1,3)(2,4),(1,3)(2,4)(5,6,7,8) ,(1,3)(2,4)(5,7)(6,8),(1,3) (2,4)(5,8,7,6),(1,4,3,2), (1,4,3,2)(5,6,7,8),(1,4,3,2) (5,7)(6,8),(1,4,3,2)(5,8,7,6)]

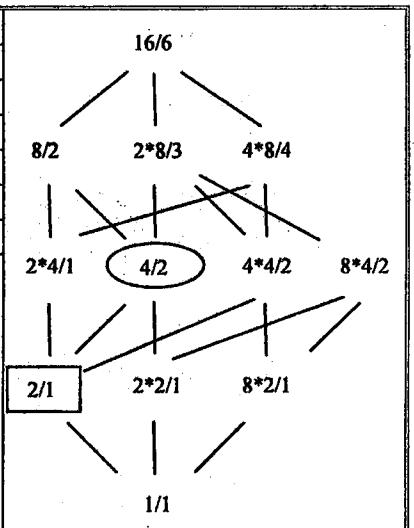


Grubun Tipi	16/4
Grubun Adı	$C_2xC_2xC_4$
GAP Tipi	16/2
GAP Adı	c4k4
Derecesi	16
Abelyenlik	Abelyen
Üreteci Elemanları	(1,2),(3,4),(5,6,7,8)
	[(),(5,6,7,8),(5,7)(6,8), (5,8,7,6),(3,4),(3,4) (5,6,7,8),(3,4)(5,7)(6,8), (3,4)(5,8,7,6),(1,2), (1,2)(5,6,7,8),(1,2)(5,7)(6,8) ,(1,2)(5,8,7,6),(1,2)(3,4), (1,2)(3,4)(5,6,7,8), (1,2)(3,4)(5,7)(6,8), (1,2)(3,4)(5,8,7,6)]

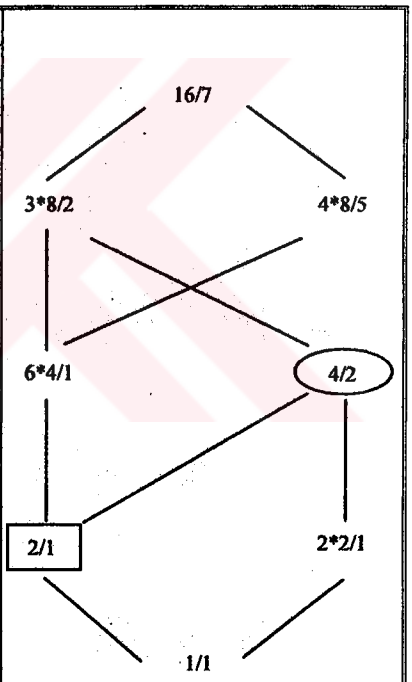


Grubun Tipi	16/5
Grubun Adı	$C_2xC_2xC_2xC_2$ - Elementary
GAP Tipi	16/1
GAP Adı	$c2^4$
Derecesi	16
Abelyenlik	Abelyen
Üreteci Elemanları	(1,2),(3,4),(5,6),(7,8)
	[(),(7,8),(5,6),(5,6)(7,8), (3,4),(3,4)(7,8),(3,4)(5,6), (3,4)(5,6)(7,8),(1,2),(1,2) (7,8),(1,2)(5,6),(1,2)(5,6) (7,8),(1,2)(3,4),(1,2)(3,4) (7,8),(1,2)(3,4)(5,6),(1,2) (3,4)(5,6)(7,8)]

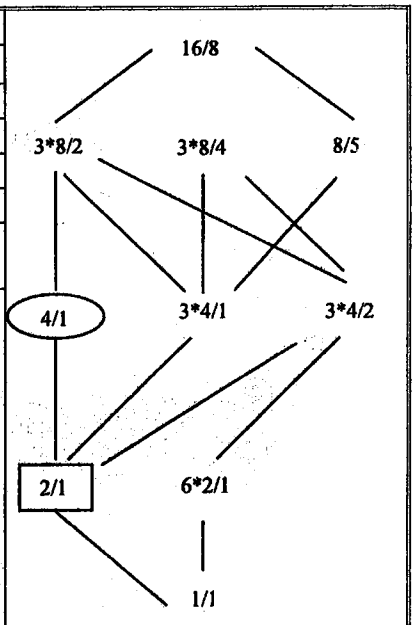
Grubun Tipi	16/6
Grubun Adı	D_4xC_2
GAP Tipi	16/6
GAP Adı	d8c2
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4),(2,4),(5,6)
Elemanları	[(),(5,6),(2,4),(2,4)(5,6), (1,2)(3,4),(1,2)(3,4)(5,6), (1,2,3,4),(1,2,3,4)(5,6), (1,3),(1,3)(5,6),(1,3)(2,4), (1,3)(2,4)(5,6),(1,4,3,2), (1,4,3,2)(5,6),(1,4)(2,3), (1,4)(2,3)(5,6)]



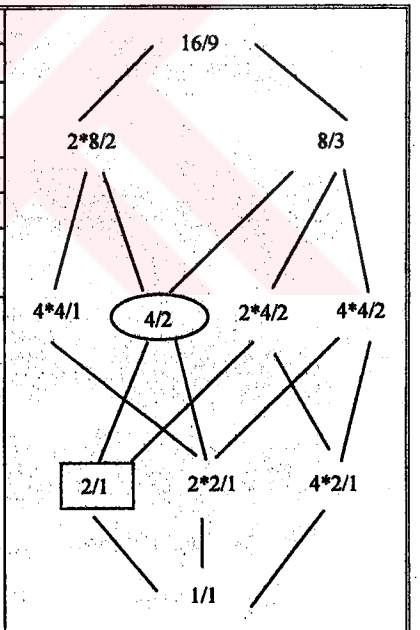
Grubun Tipi	16/7
Grubun Adı	QxC_2
GAP Tipi	16/7
GAP Adı	q8c2
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,5,2,6)(3,7,4,8), (1,7,2,8)(3,6,4,5),(9,10)
Elemanları	[(),(9,10),(1,2)(3,4)(5,6) (7,8),(1,2)(3,4)(5,6)(7,8) (9,10),(1,3,2,4)(5,8,6,7), (1,3,2,4)(5,8,6,7)(9,10), (1,4,2,3)(5,7,6,8), (1,4,2,3)(5,7,6,8)(9,10), (1,5,2,6)(3,7,4,8), (1,5,2,6)(3,7,4,8)(9,10), (1,6,2,5)(3,8,4,7), (1,6,2,5)(3,8,4,7)(9,10), (1,7,2,8)(3,6,4,5), (1,7,2,8)(3,6,4,5)(9,10), (1,8,2,7)(3,5,4,6), (1,8,2,7)(3,5,4,6)(9,10)]



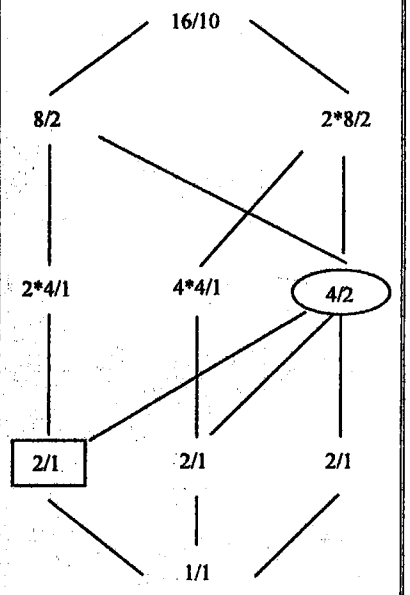
Grubun Tipi	16/8
Grubun Adı	$\Gamma_2 b$
GAP Tipi	16/8
GAP Adı	d8y4
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4)(5,6,7,8),(5,7)(6,8), (1,5)(2,6)(3,7)(4,8)
Elemanları	[(),(5,7)(6,8),(1,2,3,4) (5,6,7,8),(1,2,3,4)(5,8,7,6), (1,3)(2,4),(1,3)(2,4)(5,7) (6,8),(1,4,3,2)(5,6,7,8), (1,4,3,2)(5,8,7,6),(1,5)(2,6) (3,7)(4,8),(1,5,3,7)(2,6,4,8), (1,6,3,8)(2,7,4,5),(1,6)(2,7) (3,8)(4,5),(1,7,3,5)(2,8,4,6), (1,7)(2,8)(3,5)(4,6),(1,8) (2,5)(3,6)(4,7),(1,8,3,6) (2,5,4,7)]



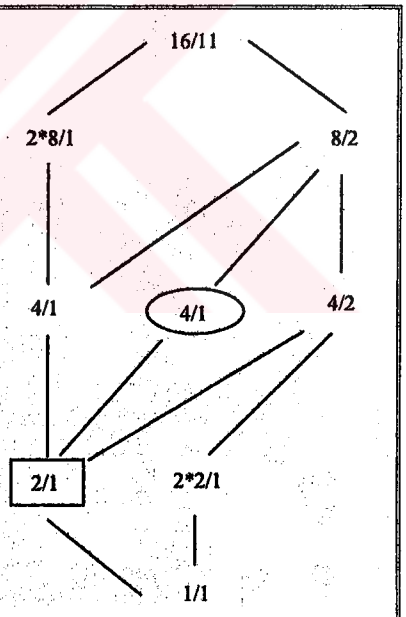
Grubun Tipi	16/9
Grubun Adı	$\Gamma_2 C_1$
GAP Tipi	16/9
GAP Adı	$c_4 c_2 \chi c_2$
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4)(5,6,7,8),(1,4,5,8) (2,7,6,3)
Elemanları	[(),(2,8)(4,6),(1,2,3,4) (5,6,7,8),(1,2,5,6)(3,4,7,8), (1,3)(2,4)(5,7)(6,8),(1,3) (2,6)(4,8)(5,7),(1,4,3,2) (5,8,7,6),(1,4,5,8)(2,7,6,3), (1,5)(2,4)(3,7)(6,8),(1,5) (2,6)(3,7)(4,8),(1,6,5,2) (3,8,7,4),(1,6,3,8)(2,7,4,5), (1,7)(3,5),(1,7)(2,8)(3,5) (4,6),(1,8,5,4)(2,3,6,7), (1,8,3,6)(2,5,4,7)]



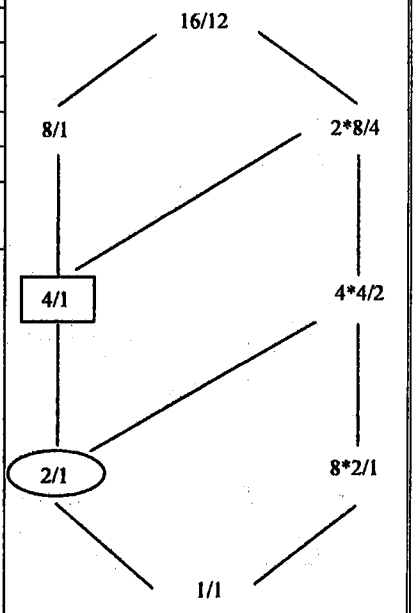
Grubun Tipi	16/10
Grubun Adı	Γ_2C_2
GAP Tipi	16/10
GAP Adı	$C_4\chi C_4$
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4),(1,3)(5,6,7,8)
Elemanları	[(),(5,7)(6,8),(2,4)(5,6,7,8), (2,4)(5,8,7,6),(1,2)(3,4) (5,6,7,8),(1,2)(3,4)(5,8,7,6), (1,2,3,4),(1,2,3,4)(5,7)(6,8), (1,3)(5,6,7,8),(1,3)(5,8,7,6), (1,3)(2,4),(1,3)(2,4)(5,7) (6,8),(1,4,3,2),(1,4,3,2)(5,7) (6,8),(1,4)(2,3)(5,6,7,8), (1,4)(2,3)(5,8,7,6)]



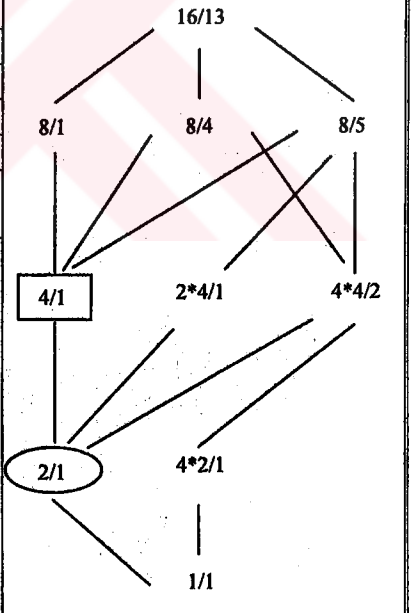
Grubun Tipi	16/11
Grubun Adı	Γ_2d
GAP Tipi	16/11
GAP Adı	$C_2\chi C_8$
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8),(2,6)(4,8)
Elemanları	[(),(2,6)(4,8), (1,2,3,4,5,6,7,8), (1,2,7,8,5,6,3,4),(1,3,5,7) (2,4,6,8), (1,3,5,7)(2,8,6,4), (1,4,3,6,5,8,7,2), (1,4,7,2,5,8,3,6),(1,5)(3,7), (1,5)(2,6)(3,7)(4,8), (1,6,7,4,5,2,3,8), (1,6,3,8,5,2,7,4), (1,7,5,3)(2,4,6,8), (1,7,5,3)(2,8,6,4), (1,8,7,6,5,4,3,2), (1,8,3,2,5,4,7,6)]



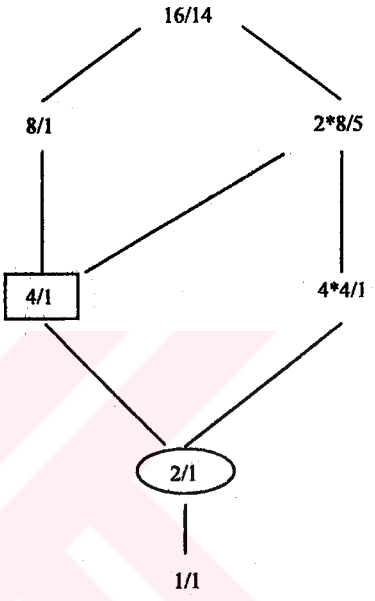
Grubun Tipi	16/12
Grubun Adı	D_8 - Dihedral
GAP Tipi	16/12
GAP Adı	d16
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6)
Elemanları	[(),(2,8)(3,7)(4,6), (1,2)(3,8)(4,7)(5,6), (1,2,3,4,5,6,7,8),(1,3)(4,8) (5,7),(1,3,5,7)(2,4,6,8),(1,4) (2,3)(5,8)(6,7), (1,4,7,2,5,8,3,6),(1,5)(2,4) (6,8),(1,5)(2,6)(3,7)(4,8), (1,6)(2,5)(3,4)(7,8), (1,6,3,8,5,2,7,4),(1,7)(2,6) (3,5),(1,7,5,3)(2,8,6,4), (1,8,7,6,5,4,3,2),(1,8)(2,7) (3,6)(4,5)]



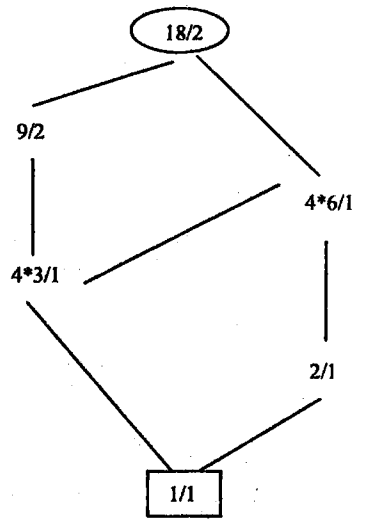
Grubun Tipi	16/13
Grubun Adı	QxD_8
GAP Tipi	16/13
GAP Adı	qd16
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8), (2,4)(3,7)(6,8)
Elemanları	[(),(2,4)(3,7)(6,8), (1,2,3,4,5,6,7,8), (1,2,5,6)(3,8,7,4), (1,3,5,7)(2,4,6,8),(1,3)(2,6) (5,7),(1,4,7,2,5,8,3,6), (1,4,5,8)(2,7,6,3),(1,5)(2,6) (3,7)(4,8),(1,5)(2,8)(4,6), (1,6,5,2)(3,4,7,8), (1,6,3,8,5,2,7,4),(1,7)(3,5) (4,8),(1,7,5,3)(2,8,6,4), (1,8,7,6,5,4,3,2), (1,8,5,4)(2,3,6,7)]



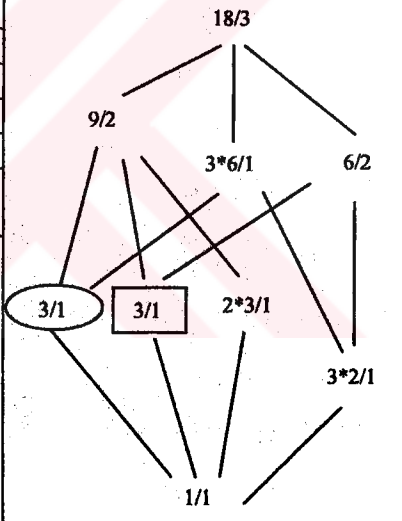
Grubun Tipi	16/14
Grubun Adı	Q_8 - Dicyclic
GAP Tipi	16/14
GAP Adı	q16
Derecesi	16
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8) (9,16,15,14,13,12,11,10), (1,9,5,13)(2,10,6,14) (3,11,7,15)(4,12,8,16)
Elemanları	[(),(1,2,3,4,5,6,7,8) (9,16,15,14,13,12,11,10), (1,3,5,7)(2,4,6,8)(9,15,13,11) (10,16,14,12), (1,4,7,2,5,8,3,6) (9,14,11,16,13,10,15,12),(1,5) (2,6)(3,7)(4,8)(9,13)(10,14)(1 1,15)(12,16),(1,6,3,8,5,2,7,4) (9,12,15,10,13,16,11,14), (1,7,5,3)(2,8,6,4)(9,11,13,15) (10,12,14,16), (1,8,7,6,5,4,3,2) (9,10,11,12,13,14,15,16), (1,9,5,13)(2,10,6,14) (3,11,7,15)(4,12,8,16), (1,10,5,14)(2,11,6,15) (3,12,7,16)(4,13,8,9), (1,11,5,15)(2,12,6,16) (3,13,7,9)(4,14,8,10), (1,12,5,16)(2,13,6,9) (3,14,7,10)(4,15,8,11), (1,13,5,9)(2,14,6,10) (3,15,7,11)(4,16,8,12), (1,14,5,10)(2,15,6,11) (3,16,7,12)(4,9,8,13), (1,15,5,11)(2,16,6,12) (3,9,7,13)(4,10,8,14), (1,16,5,12)(2,9,6,13) (3,10,7,14)(4,11,8,15)]



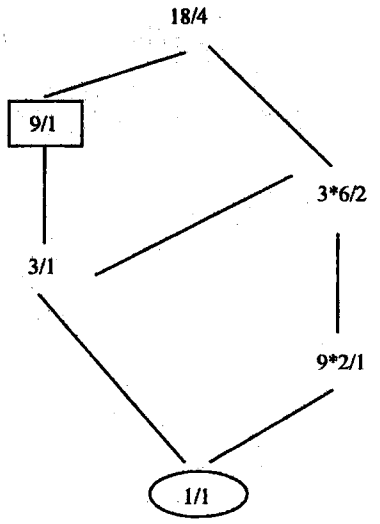
Grubun Tipi	18/2
Grubun Adı	C_6xC_3
GAP Tipi	18/1
GAP Adı	c6c3
Derecesi	18
Abelyenlik	Abelyen
Üreteci	(1,2,3)(4,5),(6,7,8)
Elemanları	[(),(6,7,8),(6,8,7),(4,5),(4,5)(6,7,8),(4,5)(6,8,7),(1,2,3),(1,2,3)(6,7,8),(1,2,3)(6,8,7),(1,2,3)(4,5),(1,2,3)(4,5)(6,7,8),(1,2,3)(4,5)(6,8,7),(1,3,2),(1,3,2)(6,7,8),(1,3,2)(6,8,7),(1,3,2)(4,5),(1,3,2)(4,5)(6,7,8),(1,3,2)(4,5)(6,8,7)]



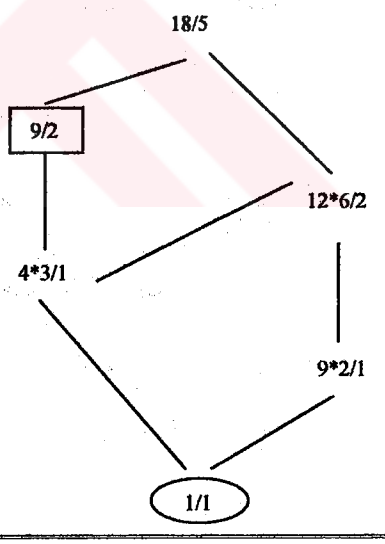
Grubun Tipi	18/3
Grubun Adı	S_3xC_3
GAP Tipi	18/4
GAP Adı	s3c3
Derecesi	18
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8),(2,4)(3,7)(6,8)
Elemanları	[(),(2,4)(3,7)(6,8),(1,2,3,4,5,6,7,8),(1,2,5,6)(3,8,7,4),(1,3,5,7)(2,4,6,8),(1,3)(2,6)(5,7),(1,4,7,2,5,8,3,6),(1,4,5,8)(2,7,6,3),(1,5)(2,6)(3,7)(4,8),(1,5)(2,8)(4,6),(1,6,5,2)(3,4,7,8),(1,6,3,8,5,2,7,4),(1,7)(3,5)(4,8),(1,7,5,3)(2,8,6,4),(1,8,7,6,5,4,3,2),(1,8,5,4)(2,3,6,7)]



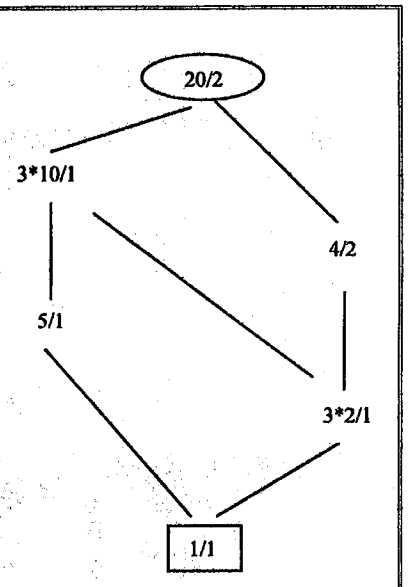
Grubun Tipi	18/4
Grubun Adı	D_9 - Dihedral
GAP Tipi	18/3
GAP Adı	d18
Derecesi	18
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8,9), (1,9)(2,8)(3,7)(4,6)
Elemanları	[(), (2,9)(3,8)(4,7)(5,6), (1,2)(3,9)(4,8)(5,7), (1,2,3,4,5,6,7,8,9), (1,3)(4,9)(5,8)(6,7), (1,3,5,7,9,2,4,6,8), (1,4)(2,3)(5,9)(6,8), (1,4,7)(2,5,8)(3,6,9), (1,5)(2,4)(6,9)(7,8), (1,5,9,4,8,3,7,2,6), (1,6)(2,5)(3,4)(7,9), (1,6,2,7,3,8,4,9,5), (1,7)(2,6)(3,5)(8,9), (1,7,4)(2,8,5)(3,9,6), (1,8)(2,7)(3,6)(4,5), (1,8,6,4,2,9,7,5,3), (1,9,8,7,6,5,4,3,2), (1,9)(2,8)(3,7)(4,6)]



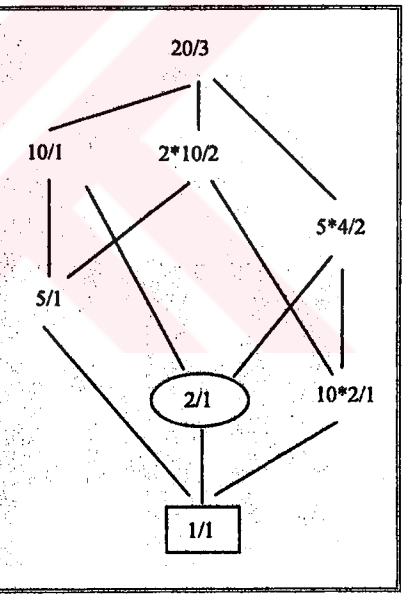
Grubun Tipi	18/5
Grubun Adı	$(C_3xC_3)x\ C_2$
GAP Tipi	18/5
GAP Adı	$c3^2xc2$
Derecesi	18
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3), (4,5,6), (2,3)(5,6)
Elemanları	[(), (4,5,6), (4,6,5), (2,3)(5,6), (2,3)(4,5), (2,3)(4,6), (1,2)(5,6), (1,2)(4,5), (1,2)(4,6), (1,2,3), (1,2,3)(4,5,6), (1,2,3)(4,6,5), (1,3,2), (1,3,2)(4,5,6), (1,3,2)(4,6,5), (1,3)(5,6), (1,3)(4,5), (1,3)(4,6)]



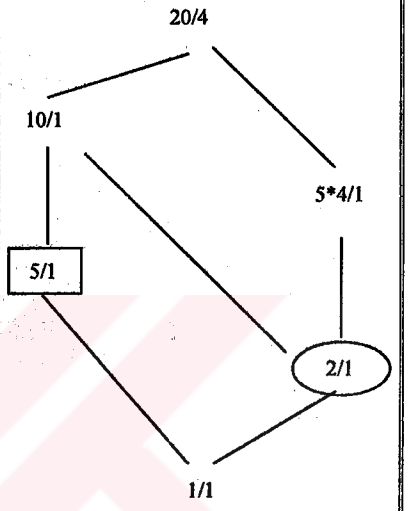
Grubun Tipi	20/2
Grubun Adı	$C_{10}xC_2$
GAP Tipi	20/1
GAP Adı	c10c2
Derecesi	20
Abelyenlik	Abelyen
Üreteci	(1,2,3,4,5)(6,7),(8,9)
Elemanları	[(),(8,9),(6,7),(6,7)(8,9), (1,2,3,4,5),(1,2,3,4,5)(8,9), (1,2,3,4,5)(6,7),(1,2,3,4,5) (6,7)(8,9),(1,3,5,2,4), (1,3,5,2,4)(8,9),(1,3,5,2,4) (6,7),(1,3,5,2,4)(6,7)(8,9), (1,4,2,5,3),(1,4,2,5,3)(8,9), (1,4,2,5,3)(6,7),(1,4,2,5,3) (6,7)(8,9),(1,5,4,3,2), (1,5,4,3,2)(8,9),(1,5,4,3,2) (6,7),(1,5,4,3,2)(6,7)(8,9)]



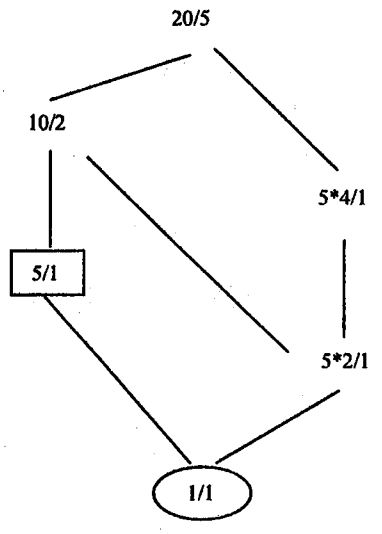
Grubun Tipi	20/3
Grubun Adı	D_{10} - Dihedral
GAP Tipi	20/3
GAP Adı	d20
Derecesi	20
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5)(6,7),(2,5)(3,4)
Elemanları	[(),(6,7),(2,5)(3,4),(2,5) (3,4)(6,7),(1,2)(3,5),(1,2) (3,5)(6,7),(1,2,3,4,5), (1,2,3,4,5)(6,7),(1,3)(4,5), (1,3)(4,5)(6,7),(1,3,5,2,4), (1,3,5,2,4)(6,7),(1,4)(2,3), (1,4)(2,3)(6,7),(1,4,2,5,3), (1,4,2,5,3)(6,7),(1,5,4,3,2), (1,5,4,3,2)(6,7),(1,5)(2,4), (1,5)(2,4)(6,7)]



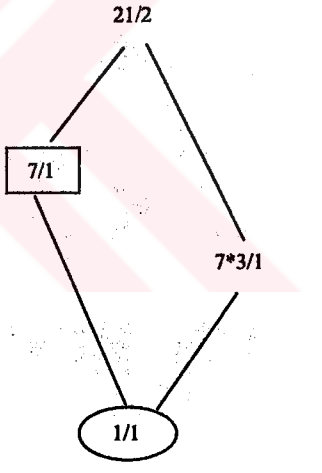
Grubun Tipi	20/4
Grubun Adı	Q_{10} - Dicyclic
GAP Tipi	20/4
GAP Adı	q20
Derecesi	20
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8,9,10) (11,20,19,18,17,16,15,14,13,12) ,(1,11,6,16)(2,12,7,17) (3,13,8,18)(4,14,9,19) (5,15,10,20)
Elemanları	[(),(1,2,3,4,5,6,7,8,9,10) (11,20,19,18,17,16,15,14,13,12),(1,3, 5,7,9)(2,4,6,8,10) (11,19,17,15,13)(12,20,18,16,14), (1,4,7,10,3,6,9,2,5,8) (11,18,15,12,19,16,13,20,17,14),(1,5, 9,3,7)(2,6,10,4,8)(11,17,13,19,15) (12,18,14,20,16),(1,6)(2,7) (3,8)(4,9)(5,10)(11,16)(12,17) (13,18)(14,19)(15,20), (1,7,3,9,5)(2,8,4,10,6) (11,15,19,13,17)(12,16,20,14,18), (1,8,5,2,9,6,3,10,7,4) (11,14,17,20,13,16,19,12,15,18),(1,9, 7,5,3)(2,10,8,6,4)(11,13,15,17,19) (12,14,16,18,20), (1,10,9,8,7,6,5,4,3,2) (11,12,13,14,15,16,17,18,19,20),(1,11 ,6,16)(2,12,7,17)(3,13,8,18) (4,14,9,19)(5,15,10,20),(1,12,6,17) (2,13,7,18)(3,14,8,19)(4,15,9,20) (5,16,10,11),(1,13,6,18)(2,14,7,19) (3,15,8,20)(4,16,9,11)(5,17,10,12), (1,14,6,19)(2,15,7,20)(3,16,8,11) (4,17,9,12)(5,18,10,13),(1,15,6,20) (2,16,7,11)(3,17,8,12)(4,18,9,13) (5,19,10,14),(1,16,6,11)(2,17,7,12) (3,18,8,13)(4,19,9,14)(5,20,10,15), (1,17,6,12)(2,18,7,13)(3,19,8,14) (4,20,9,15)(5,11,10,16),(1,18,6,13) (2,19,7,14)(3,20,8,15)(4,11,9,16) (5,12,10,17),(1,19,6,14)(2,20,7,15) (3,11,8,16)(4,12,9,17)(5,13,10,18), (1,20,6,15)(2,11,7,16)(3,12,8,17) (4,13,9,18)(5,14,10,19)]



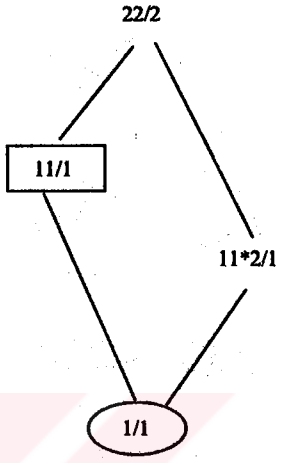
Grubun Tipi	20/5
Grubun Adı	Hol(C_5)
GAP Tipi	20/5
GAP Adı	c4xc5
Derecesi	20
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5),(2,4,5,3)
Elemanları	[(),(2,3,5,4),(2,4,5,3),(2,5)(3,4),(1,2)(3,5),(1,2,3,4,5),(1,2,4,3),(1,2,5,4),(1,3,4,2),(1,3)(4,5),(1,3,5,2,4),(1,3,2,5),(1,4,5,2),(1,4,3,5),(1,4)(2,3),(1,4,2,5,3),(1,5,4,3,2),(1,5,3,4),(1,5,2,3),(1,5)(2,4)]



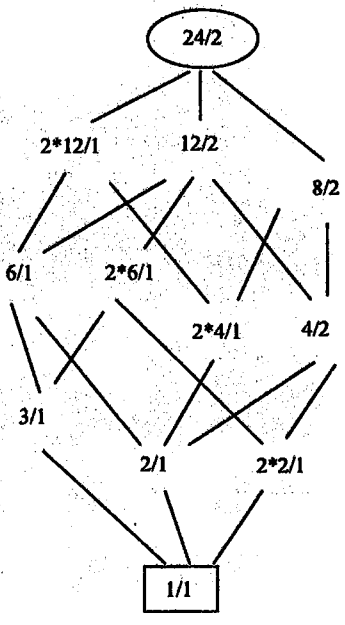
Grubun Tipi	21/2
Grubun Adı	$C_7 \chi C_3$
GAP Tipi	21/2
GAP Adı	c7xc3
Derecesi	21
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7),(2,5,3)(4,6,7)
Elemanları	[(),(2,3,5)(4,7,6),(2,5,3)(4,6,7),(1,2,3,4,5,6,7),(1,2,4)(3,6,5),(1,2,6)(4,7,5),(1,3,5,7,2,4,6),(1,3,7)(2,5,4),(1,3,4)(2,7,6),(1,4,2)(3,5,6),(1,4,7,3,6,2,5),(1,4,3)(2,6,7),(1,5,7)(3,6,4),(1,5,2,6,3,7,4),(1,5,6)(2,7,3),(1,6,2)(4,5,7),(1,6,5)(2,3,7),(1,6,4,2,7,5,3),(1,7,6,5,4,3,2),(1,7,5)(3,4,6),(1,7,3)(2,4,5)]



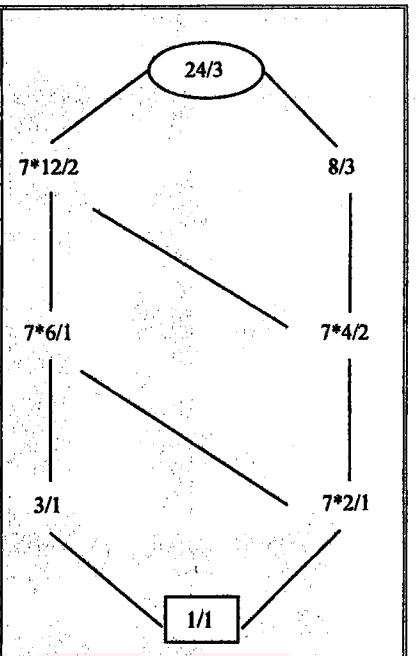
Grubun Tipi	22/2
Grubun Adı	D_{11} - Dihedral
GAP Tipi	22/2
GAP Adı	d22
Derecesi	22
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8,9,10,11), (2,11)(3,10)(4,9)(5,8)(6,7)
Elemanları	[(),(2,11)(3,10)(4,9)(5,8) (6,7),(1,2)(3,11) (4,10)(5,9)(6,8), (1,2,3,4,5,6,7,8,9,10,11), (1,3)(4,11)(5,10)(6,9)(7,8), (1,3,5,7,9,11,2,4,6,8,10), (1,4)(2,3)(5,11)(6,10)(7,9), (1,4,7,10,2,5,8,11,3,6,9), (1,5)(2,4)(6,11)(7,10)(8,9), (1,5,9,2,6,10,3,7,11,4,8), (1,6)(2,5)(3,4)(7,11)(8,10), (1,6,11,5,10,4,9,3,8,2,7), (1,7)(2,6)(3,5)(8,11)(9,10), (1,7,2,8,3,9,4,10,5,11,6), (1,8)(2,7)(3,6)(4,5)(9,11), (1,8,4,11,7,3,10,6,2,9,5), (1,9)(2,8)(3,7)(4,6)(10,11), (1,9,6,3,11,8,5,2,10,7,4), (1,10)(2,9)(3,8)(4,7)(5,6), (1,10,8,6,4,2,11,9,7,5,3), (1,11,10,9,8,7,6,5,4,3,2), (1,11)(2,10)(3,9)(4,8)(5,7)]



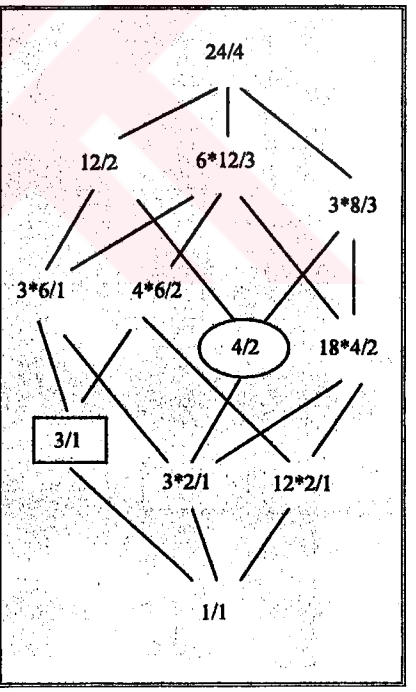
Grubun Tipi	24/2
Grubun Adı	C_2xC_{12}
GAP Tipi	24/2
GAP Adı	c12c2
Derecesi	24
Abelyenlik	Abelyen
Üreteci	(1,2),(3,4,5,6)(7,8,9)
Elemanları	[(),(7,8,9),(7,9,8),(3,4,5,6), (3,4,5,6)(7,8,9),(3,4,5,6) (7,9,8),(3,5)(4,6),(3,5)(4,6) (7,8,9),(3,5)(4,6)(7,9,8), (3,6,5,4),(3,6,5,4)(7,8,9), (3,6,5,4)(7,9,8),(1,2),(1,2) (7,8,9),(1,2)(7,9,8),(1,2) (3,4,5,6),(1,2)(3,4,5,6)(7,8,9) ,(1,2)(3,4,5,6)(7,9,8),(1,2) (3,5)(4,6),(1,2)(3,5)(4,6) (7,8,9),(1,2)(3,5)(4,6)(7,9,8), (1,2)(3,6,5,4),(1,2)(3,6,5,4)(7 ,8,9),(1,2)(3,6,5,4)(7,9,8)]



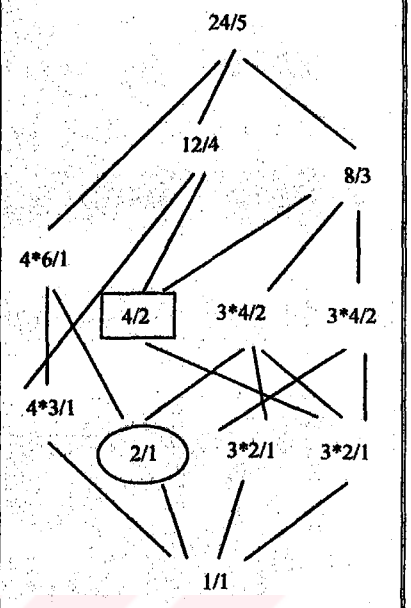
Grubun Tipi	24/3
Grubun Adı	$C_6xC_2^2$
GAP Tipi	24/1
GAP Adı	c6k4
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci Elemanları	(1,2,3)(4,5),(6,7),(8,9)
	[(),(8,9),(6,7),(6,7)(8,9),(4,5),(4,5)(8,9),(4,5)(6,7),(4,5)(6,7)(8,9),(1,2,3),(1,2,3)(8,9),(1,2,3)(6,7),(1,2,3)(6,7)(8,9),(1,2,3)(4,5),(1,2,3)(4,5)(8,9),(1,2,3)(4,5)(6,7),(1,2,3)(4,5)(6,7)(8,9),(1,3,2),(1,3,2)(8,9),(1,3,2)(6,7),(1,3,2)(6,7)(8,9),(1,3,2)(4,5),(1,3,2)(4,5)(8,9),(1,3,2)(4,5)(6,7),(1,3,2)(4,5)(6,7)(8,9)]



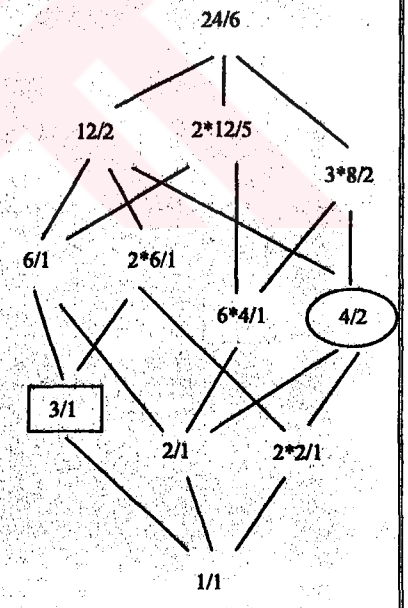
Grubun Tipi	24/4
Grubun Adı	D_6xC_2
GAP Tipi	24/6
GAP Adı	d12c2
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci Elemanları	(1,2,3)(4,5),(2,3),(6,7)
	[(),(6,7),(4,5),(4,5)(6,7),(2,3),(2,3)(6,7),(2,3)(4,5),(2,3)(4,5)(6,7),(1,2),(1,2)(6,7),(1,2)(4,5),(1,2)(4,5)(6,7),(1,2,3),(1,2,3)(6,7),(1,2,3)(4,5),(1,2,3)(4,5)(6,7),(1,3,2),(1,3,2)(6,7),(1,3,2)(4,5),(1,3,2)(4,5)(6,7),(1,3),(1,3)(6,7),(1,3)(4,5),(1,3)(4,5)(6,7)]



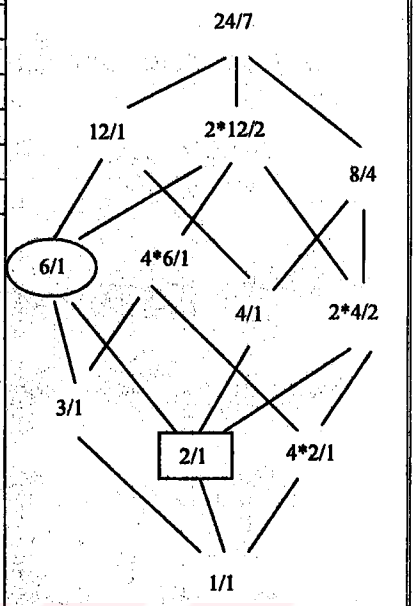
Grubun Tipi	24/5
Grubun Adı	A_4xC_2
GAP Tipi	24/10
GAP Adı	a4c2
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci Elemanları	(1,2,3),(2,3,4),(5,6)
	[(),(5,6),(2,3,4),(2,3,4)(5,6),(2,4,3),(2,4,3)(5,6),(1,2)(3,4),(1,2)(3,4)(5,6),(1,2,3),(1,2,3)(5,6),(1,2,4),(1,2,4)(5,6),(1,3,2),(1,3,2)(5,6),(1,3,4),(1,3,4)(5,6),(1,3)(2,4),(1,3)(2,4)(5,6),(1,4,2),(1,4,2)(5,6),(1,4,3),(1,4,3)(5,6),(1,4)(2,3),(1,4)(2,3)(5,6)]



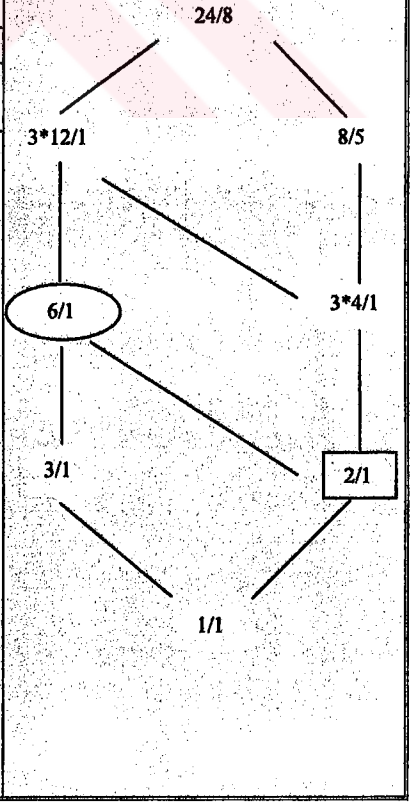
Grubun Tipi	24/6
Grubun Adı	Q_6xC_2
GAP Tipi	24/8
GAP Adı	q12c2
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci Elemanları	(1,2,3)(4,5),(2,3),(6,7)
	[(),(6,7),(4,5),(4,5)(6,7),(2,3),(2,3)(6,7),(2,3)(4,5),(2,3)(4,5)(6,7),(1,2),(1,2)(6,7),(1,2)(4,5),(1,2)(4,5)(6,7),(1,2,3),(1,2,3)(6,7),(1,2,3)(4,5),(1,2,3)(4,5)(6,7),(1,3,2),(1,3,2)(6,7),(1,3,2)(4,5),(1,3,2)(4,5)(6,7),(1,3),(1,3)(6,7),(1,3)(4,5),(1,3)(4,5)(6,7)]



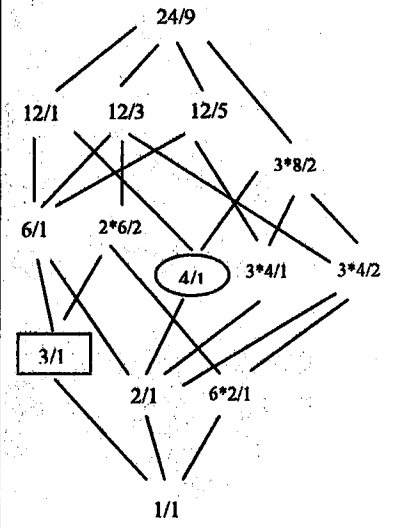
Grubun Tipi	24/7
Grubun Adı	D_4xC_3
GAP Tipi	24/4
GAP Adı	d8c3
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4)(5,6,7),(2,4)
Elemanları	[(),(5,6,7),(5,7,6),(2,4),(2,4)(5,6,7),(2,4)(5,7,6),(1,2)(3,4),(1,2)(3,4)(5,6,7),(1,2)(3,4)(5,7,6),(1,2,3,4),(1,2,3,4)(5,6,7),(1,2,3,4)(5,7,6),(1,3),(1,3)(5,6,7),(1,3)(5,7,6),(1,3)(2,4),(1,3)(2,4)(5,6,7),(1,3)(2,4)(5,7,6),(1,4,3,2),(1,4,3,2)(5,6,7),(1,4,3,2)(5,7,6),(1,4)(2,3),(1,4)(2,3)(5,6,7),(1,4)(2,3)(5,7,6)]



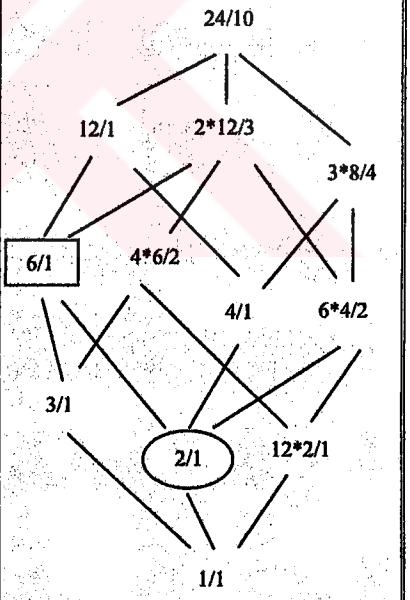
Grubun Tipi	24/8
Grubun Adı	QxC_3
GAP Tipi	24/5
GAP Adı	q8c3
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,5,2,6)(3,7,4,8),(1,7,2,8)(3,6,4,5),(9,10,11)
Elemanları	[(),(9,10,11),(9,11,10),(1,2)(3,4)(5,6)(7,8),(1,2)(3,4)(5,6)(7,8)(9,10,11),(1,2)(3,4)(5,6)(7,8)(9,11,10),(1,3,2,4)(5,8,6,7),(1,3,2,4)(5,8,6,7)(9,10,11),(1,3,2,4)(5,8,6,7)(9,11,10),(1,4,2,3)(5,7,6,8),(1,4,2,3)(5,7,6,8)(9,10,11),(1,4,2,3)(5,7,6,8)(9,11,10),(1,5,2,6)(3,7,4,8),(1,5,2,6)(3,7,4,8)(9,10,11),(1,5,2,6)(3,7,4,8)(9,11,10),(1,6,2,5)(3,8,4,7),(1,6,2,5)(3,8,4,7)(9,10,11),(1,6,2,5)(3,8,4,7)(9,11,10),(1,7,2,8)(3,6,4,5),(1,7,2,8)(3,6,4,5)(9,10,11),(1,7,2,8)(3,6,4,5)(9,11,10),(1,8,2,7)(3,5,4,6),(1,8,2,7)(3,5,4,6)(9,10,11),(1,8,2,7)(3,5,4,6)(9,11,10)]



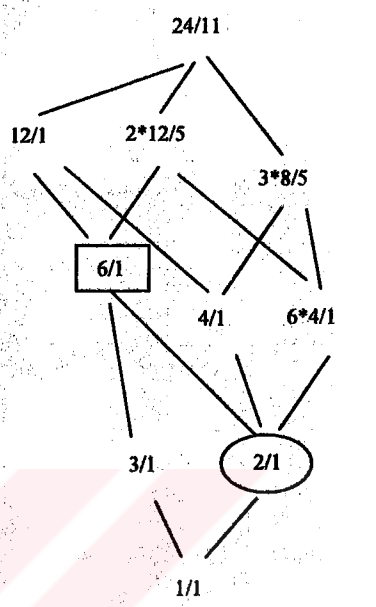
Grubun Tipi	24/9
Grubun Adı	S_3xC_4
GAP Tipi	24/7
GAP Adı	s3c4
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2),(2,3),(4,5,6,7)
Elemanları	[(),(4,5,6,7),(4,6)(5,7), (4,7,6,5),(2,3),(2,3) (4,5,6,7),(2,3)(4,6)(5,7), (2,3)(4,7,6,5),(1,2),(1,2) (4,5,6,7),(1,2)(4,6)(5,7), (1,2)(4,7,6,5),(1,2,3), (1,2,3)(4,5,6,7),(1,2,3)(4,6) (5,7),(1,2,3)(4,7,6,5),(1,3,2) ,(1,3,2)(4,5,6,7),(1,3,2)(4,6) (5,7),(1,3,2)(4,7,6,5),(1,3), (1,3)(4,5,6,7),(1,3)(4,6)(5,7) ,(1,3)(4,7,6,5)]



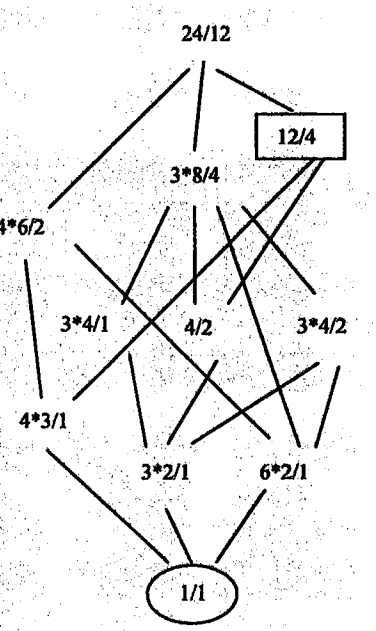
Grubun Tipi	24/10
Grubun Adı	D_{12} - Dihedral
GAP Tipi	24/12
GAP Adı	d24
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3)(4,5,6,7), (2,3)(4,7)(5,6)
Elemanları	[(),(4,5,6,7),(4,6)(5,7), (4,7,6,5),(2,3)(5,7),(2,3) (4,5)(6,7),(2,3)(4,6),(2,3) (4,7)(5,6),(1,2)(5,7),(1,2) (4,5)(6,7),(1,2)(4,6),(1,2) (4,7)(5,6),(1,2,3), (1,2,3)(4,5,6,7),(1,2,3)(4,6) (5,7),(1,2,3)(4,7,6,5), (1,3,2),(1,3,2)(4,5,6,7), (1,3,2)(4,6)(5,7),(1,3,2) (4,7,6,5),(1,3)(5,7),(1,3) (4,5)(6,7),(1,3)(4,6),(1,3) (4,7)(5,6)]



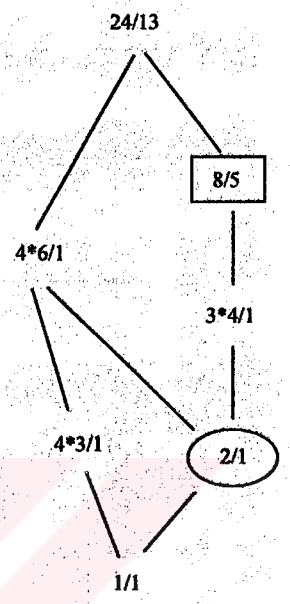
Grubun Tipi	24/11
Grubun Adı	Q_{12} - Dicyclic
GAP Tipi	24/13
GAP Adı	q24
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4)(5,6,7,8)(9,10,11), (1,5,3,7)(2,8,4,6)(10,11)
Elemanları	[(),(9,10,11),(9,11,10), (1,2,3,4)(5,6,7,8),(1,2,3,4) (5,6,7,8)(9,10,11),(1,2,3,4) (5,6,7,8)(9,11,10),(1,3)(2,4) (5,7)(6,8),(1,3)(2,4)(5,7) (6,8)(9,10,11),(1,3)(2,4)(5,7) (6,8)(9,11,10),(1,4,3,2) (5,8,7,6),(1,4,3,2)(5,8,7,6) (9,10,11),(1,4,3,2)(5,8,7,6) (9,11,10),(1,5,3,7)(2,8,4,6) (10,11),(1,5,3,7)(2,8,4,6) (9,10),(1,5,3,7)(2,8,4,6) (9,11),(1,6,3,8)(2,5,4,7) (10,11),(1,6,3,8)(2,5,4,7) (9,10),(1,6,3,8)(2,5,4,7) (9,11),(1,7,3,5)(2,6,4,8) (10,11),(1,7,3,5)(2,6,4,8) (9,10),(1,7,3,5)(2,6,4,8) (9,11),(1,8,3,6)(2,7,4,5) (10,11),(1,8,3,6)(2,7,4,5) (9,10),(1,8,3,6)(2,7,4,5) (9,11)]



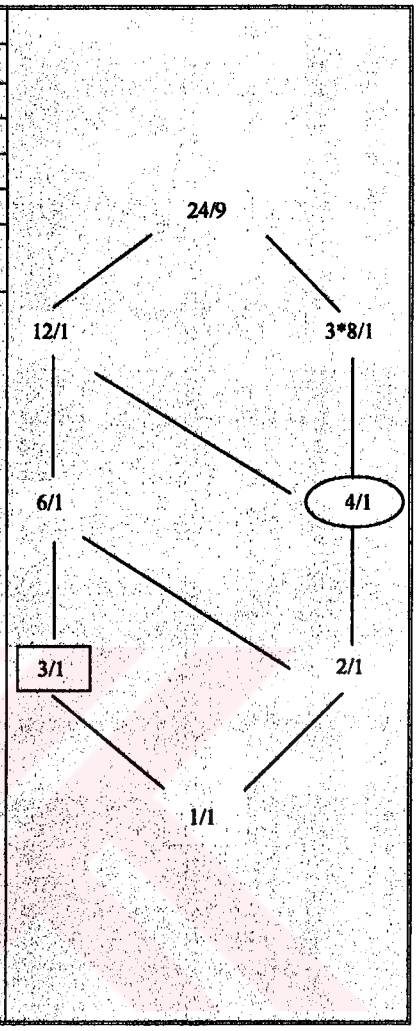
Grubun Tipi	24/12
Grubun Adı	S_4 - Simetrik
GAP Tipi	24/15
GAP Adı	s4
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3)(4,5,6,7), (2,3)(4,7)(5,6)
Elemanları	[(),(3,4),(2,3),(2,3,4),(2,4,3) ,(2,4),(1,2),(1,2)(3,4),(1,2,3) ,(1,2,3,4),(1,2,4,3),(1,2,4), (1,3,2),(1,3,4,2),(1,3),(1,3,4) ,(1,3)(2,4),(1,3,2,4),(1,4,3,2) ,(1,4,2),(1,4,3),(1,4), (1,4,2,3),(1,4)(2,3)]



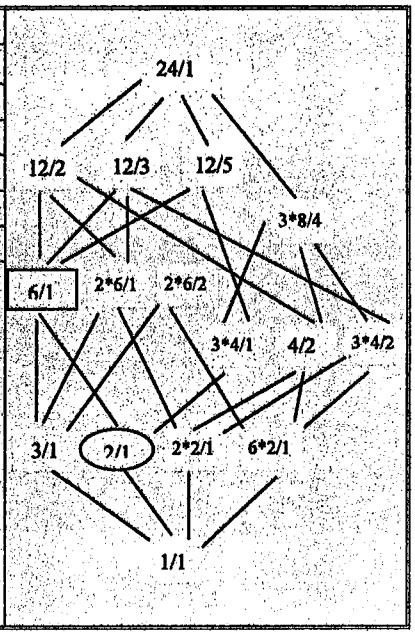
Grubun Tipi	24/13
Grubun Adı	$SL_2(F_3)$
GAP Tipi	24/14
GAP Adı	sl(2,3)
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4)(5,8,7,6), (1,5,3,7)(2,6,4,8), (2,5,6)(4,7,8)(9,10,11)
Elemanları	[(),(2,5,6)(4,7,8)(9,10,11), (2,6,5)(4,8,7)(9,11,10), (1,2,3,4)(5,8,7,6), (1,2,5,3,4,7)(6,8)(9,11,10), (1,2,8)(3,4,6)(9,10,11), (1,3)(2,4)(5,7)(6,8),(1,3) (2,7,6,4,5,8)(9,10,11),(1,3) (2,8,5,4,6,7)(9,11,10), (1,4,3,2)(5,6,7,8), (1,4,8,3,2,6)(5,7)(9,10,11), (1,4,5)(2,7,3)(9,11,10), (1,5,4)(2,3,7)(9,10,11), (1,5,6,3,7,8)(2,4)(9,11,10), (1,5,3,7)(2,6,4,8),(1,6,7) (3,8,5)(9,10,11),(1,6,2,3,8,4) (5,7)(9,11,10),(1,6,3,8) (2,7,4,5),(1,7,4,3,5,2)(6,8) (9,10,11),(1,7,6)(3,5,8) (9,11,10),(1,7,3,5)(2,8,4,6), (1,8,2)(3,6,4)(9,11,10), (1,8,7,3,6,5)(2,4)(9,10,11), (1,8,3,6)(2,5,4,7)]



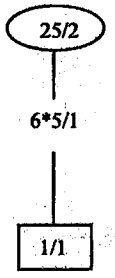
Grubun Tipi	24/14
Grubun Adı	$C_3 \chi C_8$
GAP Tipi	24/9
GAP Adı	c3xc8
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3), (2,3)(4,5,6,7,8,9,10,11)
Elemanları	[(),(4,6,8,10)(5,7,9,11), (4,8)(5,9)(6,10)(7,11), (4,10,8,6)(5,11,9,7),(2,3)(4,5, 6,7,8,9,10,11),(2,3) (4,7,10,5,8,11,6,9),(2,3) (4,9,6,11,8,5,10,7),(2,3) (4,11,10,9,8,7,6,5),(1,2) (4,5,6,7,8,9,10,11),(1,2) (4,7,10,5,8,11,6,9),(1,2) (4,9,6,11,8,5,10,7),(1,2) (4,11,10,9,8,7,6,5),(1,2,3), (1,2,3)(4,6,8,10)(5,7,9,11), (1,2,3)(4,8)(5,9)(6,10)(7,11), (1,2,3)(4,10,8,6)(5,11,9,7), (1,3,2),(1,3,2)(4,6,8,10) (5,7,9,11),(1,3,2)(4,8)(5,9) (6,10)(7,11),(1,3,2)(4,10,8,6) (5,11,9,7),(1,3) (4,5,6,7,8,9,10,11),(1,3) (4,7,10,5,8,11,6,9),(1,3) (4,9,6,11,8,5,10,7),(1,3) (4,11,10,9,8,7,6,5)]



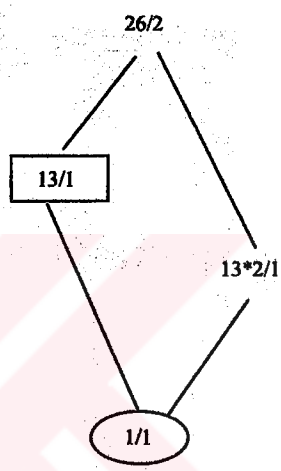
Grubun Tipi	24/15
Grubun Adı	$D_8 \chi C_3$
GAP Tipi	24/11
GAP Adı	d8xc3
Derecesi	24
Abelyenlik	Abelyen Değil
Üreteci	(5,6,7),(1,2,3,4)(6,7),(2,4)
Elemanları	[(),(5,6,7),(5,7,6),(2,4), (2,4)(5,6,7),(2,4)(5,7,6), (1,2)(3,4)(6,7),(1,2)(3,4)(5,6) ,(1,2)(3,4)(5,7),(1,2,3,4)(6,7) ,(1,2,3,4)(5,6),(1,2,3,4)(5,7), (1,3),(1,3)(5,6,7),(1,3)(5,7,6) ,(1,3)(2,4),(1,3)(2,4)(5,6,7), (1,3)(2,4)(5,7,6),(1,4,3,2) (6,7),(1,4,3,2)(5,6),(1,4,3,2) (5,7),(1,4)(2,3)(6,7),(1,4) (2,3)(5,6),(1,4)(2,3)(5,7)]



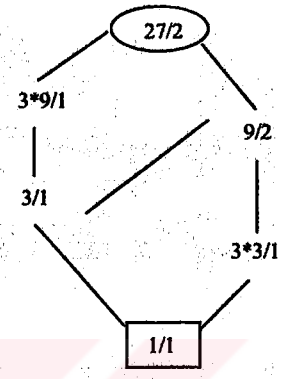
Grubun Tipi	25/2
Grubun Adı	C_5xC_5 - Elementary
GAP Tipi	25/1
GAP Adı	$c5^2$
Derecesi	25
Abelyenlik	Abelyen
Üreteci	(1,2,3,4,5),(6,7,8,9,10)
Elemanları	[(),(6,7,8,9,10),(6,8,10,7,9), (6,9,7,10,8),(6,10,9,8,7), (1,2,3,4,5),(1,2,3,4,5) (6,7,8,9,10),(1,2,3,4,5) (6,8,10,7,9),(1,2,3,4,5) (6,9,7,10,8),(1,2,3,4,5) (6,10,9,8,7),(1,3,5,2,4), (1,3,5,2,4)(6,7,8,9,10), (1,3,5,2,4)(6,8,10,7,9), (1,3,5,2,4)(6,9,7,10,8), (1,3,5,2,4)(6,10,9,8,7), (1,4,2,5,3),(1,4,2,5,3) (6,7,8,9,10),(1,4,2,5,3) (6,8,10,7,9),(1,4,2,5,3) (6,9,7,10,8),(1,4,2,5,3) (6,10,9,8,7),(1,5,4,3,2), (1,5,4,3,2)(6,7,8,9,10), (1,5,4,3,2)(6,8,10,7,9), (1,5,4,3,2)(6,9,7,10,8), (1,5,4,3,2)(6,10,9,8,7)]



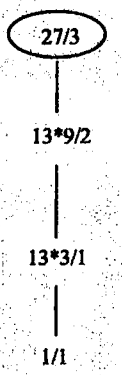
Grubun Tipi	26/2
Grubun Adı	D_{13} - Dihedral
GAP Tipi	26/2
GAP Adı	d26
Derecesi	26
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8,9,10,11,12,13) ,(2,13)(3,12)(4,11) (5,10)(6,9)(7,8)
Elemanları	[(),(2,13)(3,12)(4,11)(5,10) (6,9)(7,8),(1,2)(3,13)(4,12) (5,11)(6,10)(7,9), (1,2,3,4,5,6,7,8,9,10,11,12,13) ,(1,3)(4,13)(5,12)(6,11) (7,10)(8,9), (1,3,5,7,9,11,13,2,4,6,8,10,12) ,(1,4)(2,3)(5,13) (6,12)(7,11)(8,10), (1,4,7,10,13,3,6,9,12,2,5,8,11) ,(1,5)(2,4)(6,13)(7,12) (8,11)(9,10), (1,5,9,13,4,8,12,3,7,11,2,6,10) ,(1,6)(2,5)(3,4)(7,13) (8,12)(9,11), (1,6,11,3,8,13,5,10,2,7,12,4,9) ,(1,7)(2,6)(3,5)(8,13) (9,12)(10,11), (1,7,13,6,12,5,11,4,10,3,9,2,8) ,(1,8)(2,7)(3,6)(4,5) (9,13)(10,12), (1,8,2,9,3,10,4,11,5,12,6,13,7) ,(1,9)(2,8)(3,7)(4,6) (10,13)(11,12), (1,9,4,12,7,2,10,5,13,8,3,11,6) ,(1,10)(2,9)(3,8)(4,7) (5,6)(11,13), (1,10,6,2,11,7,3,12,8,4,13,9,5) ,(1,11)(2,10)(3,9)(4,8) (5,7)(12,13), (1,11,8,5,2,12,9,6,3,13,10,7,4) ,(1,12)(2,11)(3,10) (4,9)(5,8)(6,7), (1,12,10,8,6,4,2,13,11,9,7,5,3) ,(1,13)(2,12)(3,11)(4,10) (5,9)(6,8) (1,13,12,11,10,9,8,7,6,5,4,3,2)]



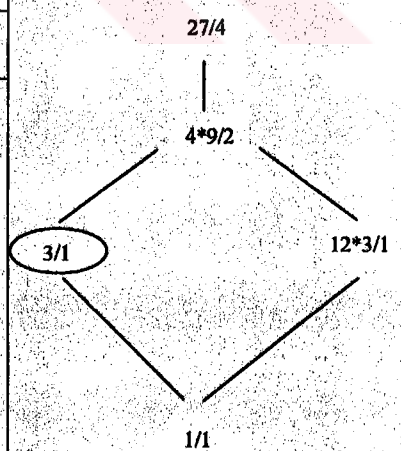
Grubun Tipi	27/2
Grubun Adı	C3xC9
GAP Tipi	27/2
GAP Adı	c9c3
Derecesi	27/2
Abelyenlik	Abelyen
Üreteci	(1,2,3),(4,5,6,7,8,9,10,11,12)
Elemanları	[(),(4,5,6,7,8,9,10,11,12), (4,6,8,10,12,5,7,9,11),(4,7,10) (5,8,11)(6,9,12), (4,8,12,7,11,6,10,5,9), (4,9,5,10,6,11,7,12,8), (4,10,7)(5,11,8)(6,12,9), (4,11,9,7,5,12,10,8,6), (4,12,11,10,9,8,7,6,5),(1,2,3), (1,2,3)(4,5,6,7,8,9,10,11,12), (1,2,3)(4,6,8,10,12,5,7,9,11), (1,2,3)(4,7,10)(5,8,11)(6,9,12) ,(1,2,3)(4,8,12,7,11,6,10,5,9), (1,2,3)(4,9,5,10,6,11,7,12,8), (1,2,3)(4,10,7)(5,11,8)(6,12,9) ,(1,2,3)(4,11,9,7,5,12,10,8,6), (1,2,3)(4,12,11,10,9,8,7,6,5), (1,3,2),(1,3,2) (4,5,6,7,8,9,10,11,12),(1,3,2) (4,6,8,10,12,5,7,9,11),(1,3,2) (4,7,10)(5,8,11)(6,9,12), (1,3,2)(4,8,12,7,11,6,10,5,9), (1,3,2)(4,9,5,10,6,11,7,12,8), (1,3,2)(4,10,7)(5,11,8) (6,12,9),(1,3,2) (4,11,9,7,5,12,10,8,6),(1,3,2) (4,12,11,10,9,8,7,6,5)]



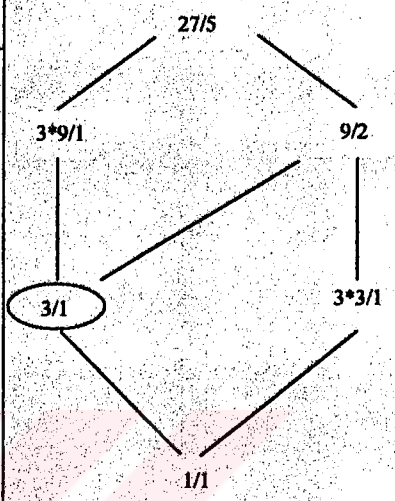
Grubun Tipi	27/3
Grubun Adı	$C_3xC_3xC_3$ - Elementary
GAP Tipi	27/1
GAP Adı	$c3^3$
Derecesi	27
Abelyenlik	Abelyen
Üreteci	(1,2,3),(4,5,6),(7,8,9)
Elemanları	[(),(7,8,9),(7,9,8),(4,5,6),(4,5,6)(7,8,9),(4,5,6)(7,9,8),(4,6,5),(4,6,5)(7,8,9),(4,6,5)(7,9,8),(1,2,3),(1,2,3)(7,8,9),(1,2,3)(7,9,8),(1,2,3)(4,5,6),(1,2,3)(4,5,6)(7,8,9),(1,2,3)(4,5,6)(7,9,8),(1,2,3)(4,6,5),(1,2,3)(4,6,5)(7,8,9),(1,2,3)(4,6,5)(7,9,8),(1,3,2),(1,3,2)(7,8,9),(1,3,2)(7,9,8),(1,3,2)(4,5,6),(1,3,2)(4,5,6)(7,8,9),(1,3,2)(4,5,6)(7,9,8),(1,3,2)(4,6,5),(1,3,2)(4,6,5)(7,8,9),(1,3,2)(4,6,5)(7,9,8)]



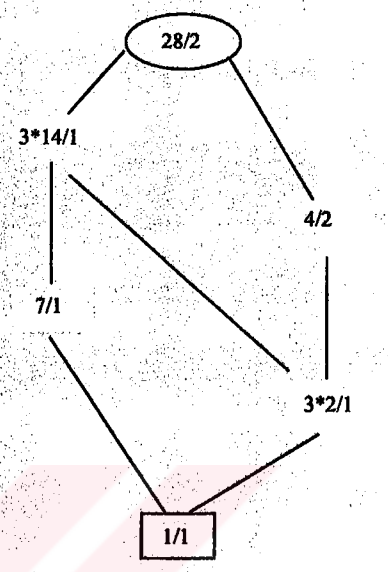
Grubun Tipi	27/4
Grubun Adı	$(C_3xC_3)\chi C_3$
GAP Tipi	27/4
GAP Adı	$c3^2xc3$
Derecesi	27
Abelyenlik	Abelyen Değil
Üreteci	(4,5,6)(7,9,8),(1,4,7)(2,5,8)(3,6,9)
Elemanları	[(),(4,5,6)(7,9,8),(4,6,5)(7,8,9),(1,2,3)(7,9,8),(1,2,3)(4,5,6)(7,8,9),(1,2,3)(4,6,5),(1,3,2)(7,8,9),(1,3,2)(4,5,6),(1,3,2)(4,6,5)(7,9,8),(1,4,7)(2,5,8)(3,6,9),(1,4,8)(2,5,9)(3,6,7),(1,4,9)(2,5,7)(3,6,8),(1,5,8)(2,6,9)(3,4,7),(1,5,9)(2,6,7)(3,4,8),(1,5,7)(2,6,8)(3,4,9),(1,6,9)(2,4,7)(3,5,8),(1,6,7)(2,4,8)(3,5,9),(1,6,8)(2,4,9)(3,5,7),(1,7,4)(2,8,5)(3,9,6),(1,7,6)(2,8,4)(3,9,5),(1,7,5)(2,8,6)(3,9,4),(1,8,4)(2,9,5)(3,7,6),(1,8,6)(2,9,4)(3,7,5),(1,8,5)(2,9,6)(3,7,4),(1,9,4)(2,7,5)(3,8,6),(1,9,6)(2,7,4)(3,8,5),(1,9,5)(2,7,6)(3,8,4)]



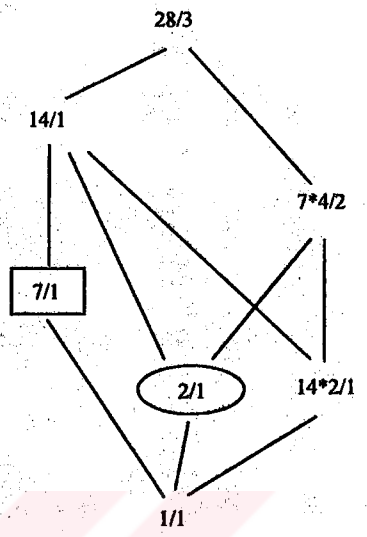
Grubun Tipi	27/5
Grubun Adı	$C_9 \chi C_3$
GAP Tipi	27/5
GAP Adı	c9xc3
Derecesi	27
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5,6,7,8,9), (2,5,8)(3,9,6)
Elemanları	[(),(2,5,8)(3,9,6),(2,8,5) (3,6,9),(1,2,3,4,5,6,7,8,9), (1,2,6,4,5,9,7,8,3), (1,2,9,4,5,3,7,8,6), (1,3,8,7,9,5,4,6,2), (1,3,5,7,9,2,4,6,8), (1,3,2,7,9,8,4,6,5), (1,4,7)(3,9,6),(1,4,7)(2,5,8) (3,6,9),1,4,7)(2,8,5), (1,5,6,4,8,9,7,2,3), (1,5,9,4,8,3,7,2,6), (1,5,3,4,8,6,7,2,9), (1,6,8,7,3,5,4,9,2), (1,6,5,7,3,2,4,9,8), (1,6,2,7,3,8,4,9,5), (1,7,4)(3,6,9),(1,7,4)(2,5,8), (1,7,4)(2,8,5)(3,9,6), (1,8,9,4,2,3,7,5,6), (1,8,3,4,2,6,7,5,9), (1,8,6,4,2,9,7,5,3), (1,9,8,7,6,5,4,3,2), (1,9,5,7,6,2,4,3,8), (1,9,2,7,6,8,4,3,5)]



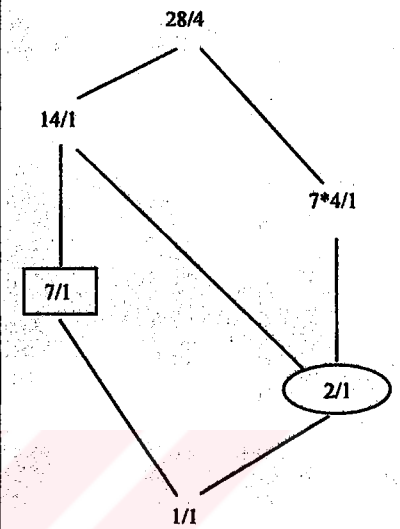
Grubun Tipi	28/2
Grubun Adı	$C_{14}xC_2$
GAP Tipi	28/1
GAP Adı	c14c2
Derecesi	28
Abelyenlik	Abelyen
Üreteci Elemanları	(1,2)(3,4,5,6,7,8,9),(10,11)
	[(),(10,11),(3,4,5,6,7,8,9), (3,4,5,6,7,8,9)(10,11), (3,5,7,9,4,6,8),(3,5,7,9,4,6,8) (10,11), (3,6,9,5,8,4,7), (3,6,9,5,8,4,7)(10,11), (3,7,4,8,5,9,6),(3,7,4,8,5,9,6) (10,11), (3,8,6,4,9,7,5), (3,8,6,4,9,7,5)(10,11), (3,9,8,7,6,5,4),(3,9,8,7,6,5,4) (10,11),(1,2),(1,2)(10,11), (1,2)(3,4,5,6,7,8,9),(1,2) (3,4,5,6,7,8,9)(10,11), (1,2)(3,5,7,9,4,6,8),(1,2) (3,5,7,9,4,6,8)(10,11),(1,2) (3,6,9,5,8,4,7),(1,2) (3,6,9,5,8,4,7)(10,11),(1,2) (3,7,4,8,5,9,6),(1,2) (3,7,4,8,5,9,6)(10,11),(1,2) (3,8,6,4,9,7,5),(1,2) (3,8,6,4,9,7,5)(10,11),(1,2) (3,9,8,7,6,5,4),(1,2) (3,9,8,7,6,5,4)(10,11)]



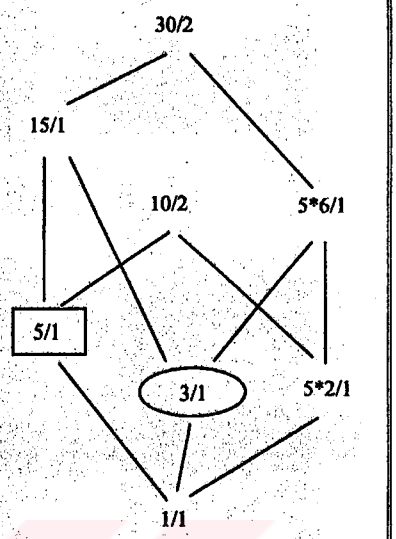
Grubun Tipi	28/3
Grubun Adı	D_{14} - Dihedral
GAP Tipi	28/3
GAP Adı	d28
Derecesi	28
Abelyenlik	Abelyen Değil
Üreteci	(1,2)(3,4,5,6,7,8,9), (3,9)(4,8)(5,7)
Elemanları	[(),(4,9)(5,8)(6,7),(3,4)(5,9) (6,8),(3,4,5,6,7,8,9),(3,5) (6,9)(7,8),(3,5,7,9,4,6,8), (3,6)(4,5)(7,9),(3,6,9,5,8,4,7) ,(3,7)(4,6)(8,9), (3,7,4,8,5,9,6),(3,8)(4,7)(5,6) ,(3,8,6,4,9,7,5), (3,9,8,7,6,5,4),(3,9)(4,8)(5,7) ,(1,2),(1,2)(4,9)(5,8)(6,7), (1,2)(3,4)(5,9)(6,8),(1,2) (3,4,5,6,7,8,9),(1,2)(3,5) (6,9)(7,8),(1,2)(3,5,7,9,4,6,8) ,(1,2)(3,6)(4,5)(7,9),(1,2) (3,6,9,5,8,4,7),(1,2)(3,7)(4,6) (8,9),(1,2)(3,7,4,8,5,9,6), (1,2)(3,8)(4,7)(5,6),(1,2) (3,8,6,4,9,7,5),(1,2)(3,9,8,7,6 ,5,4), (1,2)(3,9)(4,8)(5,7)]



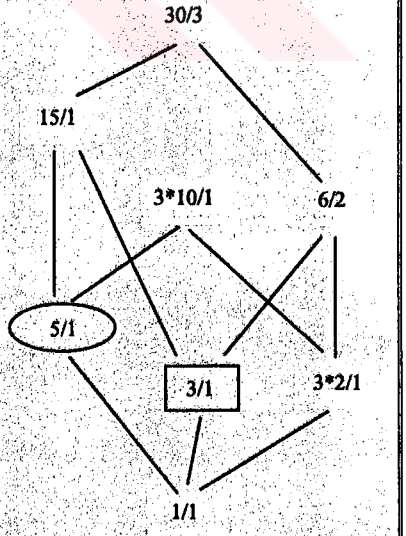
Grubun Tipi	28/4
Grubun Adı	Q_{14} - Dicyclic
GAP Tipi	28/4
GAP Adı	q28
Derecesi	28
Abelyenlik	Abelyen Değil
Üreteci	(1,2)(3,4)(5,6,7,8,9,10,11), (1,3,2,4)(6,11)(7,10)(8,9)
Elemanları	[(),(5,6,7,8,9,10,11), (5,7,9,11,6,8,10),(5,8,11,7,10, 6,9),(5,9,6,10,7,11,8), (5,10,8,6,11,9,7), (5,11,10,9,8,7,6),(1,2)(3,4), (1,2)(3,4)(5,6,7,8,9,10,11), (1,2)(3,4)(5,7,9,11,6,8,10), (1,2)(3,4)(5,8,11,7,10,6,9), (1,2)(3,4)(5,9,6,10,7,11,8), (1,2)(3,4)(5,10,8,6,11,9,7), (1,2)(3,4)(5,11,10,9,8,7,6), (1,3,2,4)(6,11)(7,10)(8,9), (1,3,2,4)(5,6)(7,11)(8,10), (1,3,2,4)(5,7)(8,11)(9,10), (1,3,2,4)(5,8)(6,7)(9,11), (1,3,2,4)(5,9)(6,8)(10,11), (1,3,2,4)(5,10)(6,9)(7,8), (1,3,2,4)(5,11)(6,10)(7,9), (1,4,2,3)(6,11)(7,10)(8,9), (1,4,2,3)(5,6)(7,11)(8,10), (1,4,2,3)(5,7)(8,11)(9,10), (1,4,2,3)(5,8)(6,7)(9,11), (1,4,2,3)(5,9)(6,8)(10,11), (1,4,2,3)(5,10)(6,9)(7,8), (1,4,2,3)(5,11)(6,10)(7,9)]



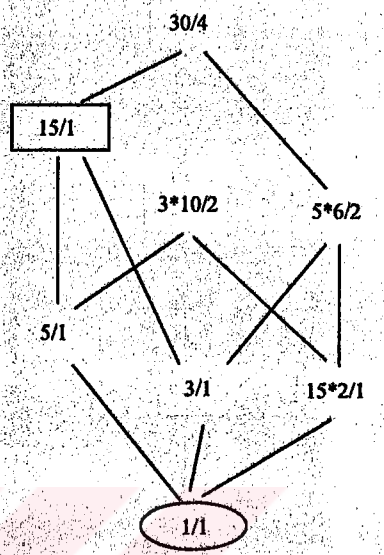
Grubun Tipi	30/2
Grubun Adı	D_5xC_3
GAP Tipi	30/2
GAP Adı	d10c3
Derecesi	30
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3,4,5),(1,5)(2,4),(6,7,8)
Elemanları	[(),(6,7,8),(6,8,7),(2,5)(3,4), (2,5)(3,4)(6,7,8),(2,5)(3,4) (6,8,7),(1,2)(3,5),(1,2)(3,5) (6,7,8),(1,2)(3,5)(6,8,7), (1,2,3,4,5),(1,2,3,4,5) (6,7,8),(1,2,3,4,5)(6,8,7), (1,3)(4,5),(1,3)(4,5)(6,7,8), (1,3)(4,5)(6,8,7),(1,3,5,2,4), (1,3,5,2,4)(6,7,8),(1,3,5,2,4) (6,8,7),(1,4)(2,3),(1,4)(2,3) (6,7,8),(1,4)(2,3)(6,8,7), (1,4,2,5,3),(1,4,2,5,3)(6,7,8), (1,4,2,5,3)(6,8,7),(1,5,4,3,2), (1,5,4,3,2)(6,7,8),(1,5,4,3,2) (6,8,7),(1,5)(2,4),(1,5)(2,4) (6,7,8),(1,5)(2,4)(6,8,7)]



Grubun Tipi	30/3
Grubun Adı	D_3xC_5
GAP Tipi	30/3
GAP Adı	d6c5
Derecesi	30
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3),(2,3),(4,5,6,7,8)
Elemanları	[(),(4,5,6,7,8),(4,6,8,5,7), (4,7,5,8,6),(4,8,7,6,5),(2,3), (2,3)(4,5,6,7,8),(2,3) (4,6,8,5,7),(2,3)(4,7,5,8,6), (2,3)(4,8,7,6,5),(1,2), (1,2)(4,5,6,7,8),(1,2) (4,6,8,5,7),(1,2)(4,7,5,8,6), (1,2)(4,8,7,6,5),(1,2,3), (1,2,3)(4,5,6,7,8),(1,2,3) (4,6,8,5,7),(1,2,3)(4,7,5,8,6), (1,2,3)(4,8,7,6,5),(1,3,2), (1,3,2)(4,5,6,7,8),(1,3,2) (4,6,8,5,7),(1,3,2)(4,7,5,8,6), (1,3,2)(4,8,7,6,5), (1,3),(1,3)(4,5,6,7,8),1,3) (4,6,8,5,7),(1,3)(4,7,5,8,6), (1,3)(4,8,7,6,5)]



Grubun Tipi	30/4
Grubun Adı	D_{15} - Dihedral
GAP Tipi	30/4
GAP Adı	d30
Derecesi	30
Abelyenlik	Abelyen Değil
Üreteci	(1,2,3)(4,5,6,7,8), (2,3)(4,8)(5,7)
Elemanları	[(), (4,5,6,7,8), (4,6,8,5,7), (4,7,5,8,6), (4,8,7,6,5), (2,3)(5,8)(6,7), (2,3)(4,5)(6,8), (2,3)(4,6)(7,8), (2,3)(4,7)(5,6), (2,3)(4,8)(5,7), (1,2)(5,8)(6,7), (1,2)(4,5)(6,8), (1,2)(4,6)(7,8), (1,2)(4,7)(5,6), (1,2)(4,8)(5,7), (1,2,3), (1,2,3)(4,5,6,7,8), (1,2,3)(4,6,8,5,7), (1,2,3)(4,7,5,8,6), (1,2,3)(4,8,7,6,5), (1,3,2), (1,3,2)(4,5,6,7,8), (1,3,2)(4,6,8,5,7), (1,3,2)(4,7,5,8,6), (1,3,2)(4,8,7,6,5), (1,3)(5,8)(6,7), (1,3)(4,5)(6,8), (1,3)(4,6)(7,8), (1,3)(4,7)(5,6), (1,3)(4,8)(5,7)]



KAYNAKLAR DİZİNİ

[1] http://www-gap.dcs.st-and.ac.uk/~gap/

[2] Arvasi, Z. ve Kocak, M., 2004, Soyut Matematik Ders Notları , 133 s. (yayımlanmamış)

[3] Arvasi, Z. , 2004, Soyut Cebire Giriş , 200 s. (yayımlanmamış)

[4] Bayraktar, M., 1998, Soyut Cebir ve Sayılar Teorisi, 271 s.

[5] Connell, E.H., 1999, Elements of Abstract and Linear Algebra, 138 p.

[6] Çallıalp, F., 1999, Sayılar Teorisi, 168 s.

[7] Çallıalp, F., 2001, Örneklerle Soyut Cebir, 300 s.

[8] Garrett, P., 1997, Intro Abstract Algebra, 200 p.

[9] Gallian, J.A., 1992, Contemporary Abstract Algebra, 525 p.

[10] Güngöroğlu, G. ve Harmancı, A., 1999, Soyut Cebire Giriş Dersleri Problem ve Çözümleri, 170 s.

[11] Hall B.C., 2000, An Elementary Introduction to Groups and Representations, 122 p.

[12] Harmancı, A., 1987, Cebir I, 247 s.

[13] Harmancı, A., 1987, Cebir II, 163 s.

[14] Hungerford, T.W., 1974, Algebra, 502 p.

[15] Judson, T.W., 1993, Abstract Algebra Theory and Applications, 427 p.

[16] Rainbolt J.G. and Gallian J.A., 2003, Abstract Algebra With GAP, 86 p. (unpublished)

[17] The GAP Group, 2002, GAP Release 4.3 Reference Manual, 875 p. (unpublished)

[18] Thomas, A. D. and Wood G. V., 1980, Group Tables 192 p.

[19] Özer, O., Çoker, D. ve Taş, K., 1999, Soyut Matematik 319 s.

[20] Şenkon, H., 1993 , Soyut Cebir Dersleri Cilt I 236 s.

DİZİN

D

E

F

G

H

I

I

K

L

M

N

O

Ö

P

Q

R

S

T

U

V

W

X

Y

Z