DEMONSTRATIONS OF USE OF MAGMA

given by Marston Conder
at Algebraic Graph Theory Summer School, Rogla (Slovenia) June 2011

```
////////////////////////////////////////////////////////////////////////
// Lecture 1 ///////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
```

```
......................................................................
Demonstration 1:  Use of MAGMA to create and analyse graphs
......................................................................
```

```
//////////////////////
// Petersen graph  //
//////////////////////
```

**** Magma input ****

```
V:= SetToSequence(Subsets({1..5},2));
E:= {{s,t}: s,t in [1..#V] | #(V[s] meet V[t]) eq 0};
P:= Graph<#V|E>;

print P;
print IsConnected(P);
print IsBipartite(P);
print Diameter(P);
print Girth(P);
print IsPlanar(P);

CliqueNumber(P);
IndependenceNumber(P);
ChromaticNumber(P);

M:=AdjacencyMatrix(P); print M;
MinimalPolynomial(M);
Eigenvalues(M);

A:=AutomorphismGroup(P); print A;
IsTransitive(P);
IsEdgeTransitive(P);
IsSymmetric(P);
IsDistanceTransitive(P);
IsDistanceRegular(P);
IsPrimitive(P);
```

**** Magma output ****

```
Magma V2.17-7     Fri Jul  1 2011 03:17:53 on jones    [Seed = 2311803970]
Type ? for help.  Type <Ctrl>-D to quit.
>
> V:= SetToSequence(Subsets({1..5},2));
> E:= {{s,t}: s,t in [1..#V] | #(V[s] meet V[t]) eq 0};
> P:= Graph<#V|E>;
>
> print P;
Graph
Vertex  Neighbours

1       3 5 6 ;
2       6 8 9 ;
3       1 4 9 ;
4       3 8 10 ;
```

```
5       1 7 8 ;
6       1 2 10 ;
7       5 9 10 ;
8       2 4 5 ;
9       2 3 7 ;
10      4 6 7 ;

> print IsConnected(P);
true
> print IsBipartite(P);
false
> print Diameter(P);
2
> print Girth(P);
5
> print IsPlanar(P);
false
>
> CliqueNumber(P);
2
> IndependenceNumber(P);
4
> ChromaticNumber(P);
3
>
> M:=AdjacencyMatrix(P); print M;
[0 0 1 0 1 1 0 0 0 0]
[0 0 0 0 0 1 0 1 1 0]
[1 0 0 1 0 0 0 0 1 0]
[0 0 1 0 0 0 0 1 0 1]
[1 0 0 0 0 0 1 1 0 0]
[1 1 0 0 0 0 0 0 0 1]
[0 0 0 0 1 0 0 0 1 1]
[0 1 0 1 1 0 0 0 0 0]
[0 1 1 0 0 0 1 0 0 0]
[0 0 0 1 0 1 1 0 0 0]
> MinimalPolynomial(M);
$.1^3 - 2*$.1^2 - 5*$.1 + 6
> Eigenvalues(M);
{ <-2, 4>, <1, 5>, <3, 1> }
>
> A:=AutomorphismGroup(P); print A;
Permutation group A acting on a set of cardinality 10
Order = 120 = 2^3 * 3 * 5
    (2, 7)(5, 6)(8, 10)
    (2, 8)(4, 9)(5, 6)(7, 10)
    (3, 5)(4, 7)(8, 9)
    (1, 2, 3, 6, 9)(4, 10, 7, 5, 8)
> IsTransitive(P);
true
> IsEdgeTransitive(P);
true
> IsSymmetric(P);
true
> IsDistanceTransitive(P);
true
> IsDistanceRegular(P);
true
> IsPrimitive(P);
true
>
> quit;

Total time: 0.440 seconds, Total memory usage: 9.53MB


//////////////////
// Graph K_{20,11} //
```

```
//////////////////

**** Magma input ****

E:= {{s,20+t}: s in [1..20],t in [1..11] };
K:= Graph<20+11|E>;

print K;

print IsConnected(K);
print IsBipartite(K);

MinimumDegree(K);  MaximumDegree(K);

Diameter(K);  Girth(K);
print IsPlanar(K);

SpanningTree(K);

CliqueNumber(K);
IndependenceNumber(K);
ChromaticNumber(K);

M:=AdjacencyMatrix(K); print M;
MinimalPolynomial(M);
Eigenvalues(M);
Spectrum(K);

A:=AutomorphismGroup(K);
print Order(A);
print CompositionFactors(A);
IsTransitive(K);
IsEdgeTransitive(K);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 03:56:27 on jones    [Seed = 2951576943]
Type ? for help.  Type <Ctrl>-D to quit.
>
> E:= {{s,20+t}: s in [1..20],t in [1..11] };
> K:= Graph<20+11|E>;
>
> print K;
Graph
Vertex  Neighbours

1       21 22 23 24 25 26 27 28 29 30 31 ;
2       21 22 23 24 25 26 27 28 29 30 31 ;
3       21 22 23 24 25 26 27 28 29 30 31 ;
4       21 22 23 24 25 26 27 28 29 30 31 ;
5       21 22 23 24 25 26 27 28 29 30 31 ;
6       21 22 23 24 25 26 27 28 29 30 31 ;
7       21 22 23 24 25 26 27 28 29 30 31 ;
8       21 22 23 24 25 26 27 28 29 30 31 ;
9       21 22 23 24 25 26 27 28 29 30 31 ;
10      21 22 23 24 25 26 27 28 29 30 31 ;
11      21 22 23 24 25 26 27 28 29 30 31 ;
12      21 22 23 24 25 26 27 28 29 30 31 ;
13      21 22 23 24 25 26 27 28 29 30 31 ;
14      21 22 23 24 25 26 27 28 29 30 31 ;
15      21 22 23 24 25 26 27 28 29 30 31 ;
16      21 22 23 24 25 26 27 28 29 30 31 ;
17      21 22 23 24 25 26 27 28 29 30 31 ;
18      21 22 23 24 25 26 27 28 29 30 31 ;
19      21 22 23 24 25 26 27 28 29 30 31 ;
20      21 22 23 24 25 26 27 28 29 30 31 ;
21      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
22      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
```

```
23      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
24      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
25      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
26      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
27      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
28      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
29      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
30      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
31      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;

>
> print IsConnected(K);
true
> print IsBipartite(K);
true
>
> MinimumDegree(K);  MaximumDegree(K);
11 1
20 21
>
> Diameter(K);  Girth(K);
2
4
> print IsPlanar(K);
false
>
> SpanningTree(K);
Graph
Vertex  Neighbours

1       21 22 23 24 25 26 27 28 29 30 31 ;
2       22 ;
3       22 ;
4       22 ;
5       22 ;
6       22 ;
7       22 ;
8       22 ;
9       22 ;
10      22 ;
11      22 ;
12      22 ;
13      22 ;
14      22 ;
15      22 ;
16      22 ;
17      22 ;
18      22 ;
19      22 ;
20      22 ;
21      1 ;
22      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ;
23      1 ;
24      1 ;
25      1 ;
26      1 ;
27      1 ;
28      1 ;
29      1 ;
30      1 ;
31      1 ;

>
> CliqueNumber(K);
2
> IndependenceNumber(K);
20
> ChromaticNumber(K);
```

```
2
>
> M:=AdjacencyMatrix(K); print M;
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
> MinimalPolynomial(M);
$.1^3 - 220*$.1
> Eigenvalues(M);
{ <0, 29> }
> Spectrum(K);
[ <0, 29>, <-14.8323969741913258974227948816014261219598086381950031974632,
1>,
<14.8323969741913258974227948816014261219598086381950031974632, 1> ]
>
> A:=AutomorphismGroup(K);
> print Order(A);
97113662879985303552000000
> print CompositionFactors(A);
    G
    |  Cyclic(2)
    *
    |  Alternating(11)
    *
    |  Cyclic(2)
    *
    |  Alternating(20)
    1
> IsTransitive(K);
false
> IsEdgeTransitive(K);
true
>
> quit;

Total time: 0.430 seconds, Total memory usage: 9.53MB


////////////////////////////////
// Random graph of order 2011 //
```

```
////////////////////////////////

**** Magma input ****

E:={};
for s in [1..2011] do for t in [(s+1)..2011] do
  rv:=Random([1..1000000]);
  if rv le 500000 then Include(~E,{s,t});end if;
  end for;end for;
R:=Graph<2011|E>;


print IsConnected(R);
print IsBipartite(R);

print IsRegular(R);
MinimumDegree(R);  MaximumDegree(R);
Diameter(R);  Girth(R);

A:=AutomorphismGroup(R);
print Order(A);

print IsPlanar(R);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 03:57:19 on jones    [Seed = 3153684751]
Type ? for help.  Type <Ctrl>-D to quit.
>
> E:={};
> for s in [1..2011] do for t in [(s+1)..2011] do
for|for>   rv:=Random([1..1000000]);
for|for>   if rv le 500000 then Include(~E,{s,t});end if;
for|for>   end for;end for;
> R:=Graph<2011|E>;
>
>
> print IsConnected(R);
true
> print IsBipartite(R);
false
>
> print IsRegular(R);
false
> MinimumDegree(R);  MaximumDegree(R);
931 1515
1088 408
> Diameter(R);  Girth(R);
2
3
>
> A:=AutomorphismGroup(R);
> print Order(A);
1
>
> print IsPlanar(R);
false
>
> quit;

Total time: 154.560 seconds, Total memory usage: 424.94MB


////////////////////////////////////
// Random cubic graph of order 44 //
////////////////////////////////////

**** Magma input ****
```

```
E:={};
deg:=[0 : v in [1..44]];
remvs:={1..44};
while #[pr : pr in Subsets(remvs,2) | not(pr in E)] gt 0 do
  x:=Random(remvs); y:=Random(remvs diff {x});
  if not({x,y} in E) then
    Include(~E,{x,y});
    deg[x]:=deg[x]+1; if (deg[x] eq 3) then Exclude(~remvs,x);end if;
    deg[y]:=deg[y]+1; if (deg[y] eq 3) then Exclude(~remvs,y);end if;
    end if;
  end while;
print #remvs,#E;

C:=Graph<44|E>;

print IsConnected(C);
print IsBipartite(C);
IsRegular(C),MinimumDegree(C),MaximumDegree(C);
Diameter(C);  Girth(C);

A:=AutomorphismGroup(C);
print Order(A);

CliqueNumber(C);
IndependenceNumber(C);
MaximumIndependentSet(C);
IsPlanar(C);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 03:58:44 on jones    [Seed = 3036047339]
Type ? for help.  Type <Ctrl>-D to quit.
>
> E:={};
> deg:=[0 : v in [1..44]];
> remvs:={1..44};
> while #[pr : pr in Subsets(remvs,2) | not(pr in E)] gt 0 do
while>   x:=Random(remvs); y:=Random(remvs diff {x});
while>   if not({x,y} in E) then
while|if>     Include(~E,{x,y});
while|if>     deg[x]:=deg[x]+1; if (deg[x] eq 3) then Exclude(~remvs,x);end
if\
;
while|if>     deg[y]:=deg[y]+1; if (deg[y] eq 3) then Exclude(~remvs,y);end
if\
;
while|if>     end if;
while>   end while;
> print #remvs,#E;
0 66
>
> C:=Graph<44|E>;
>
> print IsConnected(C);
true
> print IsBipartite(C);
false
> IsRegular(C),MinimumDegree(C),MaximumDegree(C);
true 3 1 3 1
> Diameter(C);  Girth(C);
7
3
>
> A:=AutomorphismGroup(C);
> print Order(A);
1
>
```

```
> CliqueNumber(C);
3
> IndependenceNumber(C);
19
> MaximumIndependentSet(C);
{ 29, 1, 31, 4, 34, 7, 9, 38, 10, 39, 40, 42, 44, 21, 23, 24, 25, 26, 28 }
> IsPlanar(C);
false
>
> quit;

Total time: 0.679 seconds, Total memory usage: 9.53MB
```

```
.....................................................................
Demonstration 2:  Use of small groups database to generate Cayley graphs
.....................................................................


////////////////////////////////////
// Cayley graphs of order 20 to 24 //
////////////////////////////////////

**** Magma input ****

for n in [20..24] do print " ";
  graphs:=[];
  tot:=NumberOfSmallGroups(n);
  print "Groups of order",n,"   Total number of groups =",tot;
  for t in [1..tot] do
    G := SmallGroup(n,t);
    elts:=[xx : xx in G];
    mt:=[[Index(elts,elts[i]*elts[j]) : i in [1..n]] : j in [1..n]];
    invs:=[Index(elts,elts[j]^-1) : j in [1..n]];
    invols:={j : j in [1..n] | (elts[j] ne Id(G)) and (invs[j] eq j)};
    otherreps:={j : j in [1..n] | invs[j] gt j};
    for S in Subsets(invols,3) do if Order(sub<G|[elts[j] : j in S]>) eq n
    then
       edges:={{i,mt[i][j]} : i in [1..n], j in S};  cgs:=Graph<n|edges>;
       okm:=false;
       for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=true;break
       gr;end if;end for;
       if not(okm) then Append(~graphs,cgs); print "Group",t,"gives Cayley
       graph of order",Order(cgs),

         "valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs),"diameter
         ",Diameter(cgs);
         end if;
       end if;end for;
    for x in invols do for y in otherreps do S:={x,y,invs[y]};
      if Order(sub<G|[elts[j] : j in S]>) eq n then
       edges:={{i,mt[i][j]} : i in [1..n], j in S};  cgs:=Graph<n|edges>;
       okm:=false;
       for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=true;break
       gr;end if;end for;
       if not(okm) then Append(~graphs,cgs); print "Group",t,"gives Cayley
       graph of order",Order(cgs),

         "valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs),"diameter
         ",Diameter(cgs);
         end if;
       end if;end for;end for;
    end for;
  end for;

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 03:59:23 on jones    [Seed = 2294963128]
```

```
Type ? for help.  Type <Ctrl>-D to quit.
>
> for n in [20..24] do print " ";
for>   graphs:=[];
for>   tot:=NumberOfSmallGroups(n);
for>   print "Groups of order",n,"   Total number of groups =",tot;
for>   for t in [1..tot] do
for|for>     G := SmallGroup(n,t);
for|for>     elts:=[xx : xx in G];
for|for>     mt:=[[Index(elts,elts[i]*elts[j]) : i in [1..n]] : j in
[1..n]]; \

for|for>     invs:=[Index(elts,elts[j]^-1) : j in [1..n]];
for|for>     invols:={j : j in [1..n] | (elts[j] ne Id(G)) and (invs[j] eq
j)}\
;
for|for>     otherreps:={j : j in [1..n] | invs[j] gt j};
for|for>     for S in Subsets(invols,3) do if Order(sub<G|[elts[j] : j in
S]>)\
 eq n then
for|for|for|if>        edges:={{i,mt[i][j]} : i in [1..n], j in S};  cgs:=
Grap\
h<n|edges>;   okm:=false;
for|for|for|if>        for gr in graphs do if IsIsomorphic(cgs,gr) then
okm:=t\
rue;break gr;end if;end for;
for|for|for|if>        if not(okm) then Append(~graphs,cgs); print
"Group",t,"\
gives Cayley graph of order",Order(cgs),
for|for|for|if|if|print>
"valency",Degree(Rep(VertexSet(cgs))),"girth\
",Girth(cgs),"diameter",Diameter(cgs);
for|for|for|if|if>          end if;
for|for|for|if>        end if;end for;
for|for>     for x in invols do for y in otherreps do S:={x,y,invs[y]};
for|for|for|for>       if Order(sub<G|[elts[j] : j in S]>) eq n then
for|for|for|for|if>        edges:={{i,mt[i][j]} : i in [1..n], j in S};
cgs:=\
Graph<n|edges>;  okm:=false;
for|for|for|for|if>        for gr in graphs do if IsIsomorphic(cgs,gr) then
ok\
m:=true;break gr;end if;end for;
for|for|for|for|if>        if not(okm) then Append(~graphs,cgs); print
"Group"\
,t,"gives Cayley graph of order",Order(cgs),
for|for|for|for|if|if|print>
"valency",Degree(Rep(VertexSet(cgs))),"g\
irth",Girth(cgs),"diameter",Diameter(cgs);
for|for|for|for|if|if>          end if;
for|for|for|for|if>        end if;end for;end for;
for|for>     end for;
for>   end for;

Groups of order 20    Total number of groups = 5
Group 2 gives Cayley graph of order 20 valency 3 girth 4 diameter 5
Group 3 gives Cayley graph of order 20 valency 3 girth 4 diameter 4
Group 4 gives Cayley graph of order 20 valency 3 girth 4 diameter 5
Group 4 gives Cayley graph of order 20 valency 3 girth 4 diameter 6
Group 4 gives Cayley graph of order 20 valency 3 girth 6 diameter 4

Groups of order 21    Total number of groups = 2

Groups of order 22    Total number of groups = 2
Group 1 gives Cayley graph of order 22 valency 3 girth 4 diameter 6
Group 1 gives Cayley graph of order 22 valency 3 girth 6 diameter 5
Group 1 gives Cayley graph of order 22 valency 3 girth 4 diameter 6

Groups of order 23    Total number of groups = 1
```

```
Groups of order 24    Total number of groups = 15
Group 2 gives Cayley graph of order 24 valency 3 girth 4 diameter 6
Group 5 gives Cayley graph of order 24 valency 3 girth 6 diameter 4
Group 5 gives Cayley graph of order 24 valency 3 girth 4 diameter 6
Group 6 gives Cayley graph of order 24 valency 3 girth 6 diameter 5
Group 6 gives Cayley graph of order 24 valency 3 girth 6 diameter 5
Group 6 gives Cayley graph of order 24 valency 3 girth 4 diameter 7
Group 6 gives Cayley graph of order 24 valency 3 girth 6 diameter 5
Group 12 gives Cayley graph of order 24 valency 3 girth 4 diameter 4
Group 12 gives Cayley graph of order 24 valency 3 girth 6 diameter 4
Group 12 gives Cayley graph of order 24 valency 3 girth 4 diameter 6
Group 12 gives Cayley graph of order 24 valency 3 girth 3 diameter 6
>
> quit;

Total time: 0.850 seconds, Total memory usage: 11.19MB


.....................................................................
Demonstration 3:  Finding Hamilton cycles in cubic graphs
.....................................................................


////////////////////////////////////////////
// Hamilton cycles in a Cayley graph for A_6 //
////////////////////////////////////////////

**** Magma input ****

procedure FindHamiltonCycle(gr);
//
// *** This procedure is due to Nick Wormald (in the 1980s) ***
//
maxverts := 12000;
maxdegree := 6;
dvec:=[ 0 : i in [1..maxdegree]];
adjm := [dvec: j in [1..maxverts]];
path := [ 0 : j in [1..maxverts]];
//
newpath:=path;
member := [true: j in [1..maxverts]];
ignore:=1;degree:=3;
//
n:=Order(gr);
vts:=VertexSet(gr);
edges:={{i,j} : i in [1..n], j in [1..n] | vts!i in Neighbours(vts!j)};
//
for i := 1 to n do
  member[i] := false;  j:=0;
  for k := 1 to n do if ({i,k} in edges) then j:=j+1;adjm[i][j]:=k;end
  if;end for;
  end for;
//Now checking consistency of data
print "Checking consistency of data";
good := true;
for i := 1 to n do for j := 1 to degree do
  k := adjm[i][j]; test := 1;
  while (adjm[k][test] ne i) do
    if (test lt degree) then test := test + 1; else good := false;print
    "Data inconsistent"; end if;
    end while;
  end for;end for;
if good then print " ... OK";end if;
//
length := 2;
path[1] := 1;path[2] := adjm[1][1];
member[path[1]] := true;member[path[2]] := true;
```

```
done := false;
counter := 0;bigcount := 0;longest := 0; longer := false;
while good and not(done) do
  counter := counter + 1;
  if ((counter div 10) * 10 eq counter) then // reverse path
    for i := 1 to length do newpath[i] := path[length + 1 - i]; end for;
    for i := 1 to length do path[i] := newpath[i]; end for;
    end if;
  if (counter gt 600000) then
    bigcount := bigcount + 1;counter := 1;
    for i := 1 to n do member[i] := false;end for;
    length := 2;path[1] := 1;path[2] := adjm[1][1];
    member[path[1]] := true;member[path[2]] := true;
    end if;
  if ((counter div 1000) * 1000 eq counter) then
    for i := 1 to 5 do member[path[length + 1 - i]] := false;end for;
    length := length - 5;
    end if;
//  extend path
longer := false;
for i := 1 to degree do if not(longer) then
  if not(member[adjm[path[length]][i]]) then longer := true;
    j := length + 1;path[j] := adjm[path[length]][i];
    length := j;member[path[j]] := true;
    end if;
  end if;end for;
//
  if (length gt longest) then longest := length; end if;
  if ((counter div 100) * 100 eq counter) then
    print "count =", counter, ", length =", length, ", longest =", longest;
    end if;
  if not(longer) then
//    switch paths
repeat
  r := Random([1..1000000]);
  new := ((degree * (r-1)) div 1000000)+1;
  newvert := adjm[path[length], new];
 until newvert ne path[length - 1];
reversing := false;
for i := 1 to length do
  if (path[i] eq newvert) then reversing := true; pivot := i; end if;
  if reversing then
    if (i eq pivot) then newpath[i] := path[i];
      else newpath[i] := path[length + 1 + pivot - i];
      end if;
   else newpath[i] := path[i];
   end if;
  end for;
for i := 1 to length do path[i] := newpath[i];end for;
//
    end if;
  if (length eq n) then
    for i := 1 to degree do done := (path[n] eq adjm[path[1]][i]); end for;
    end if;
  end while;
//
// Now doing a final check of cycle
//
// (First test the adjacencies)
for i := 1 to n - 1 do
  j := path[i];k := path[i + 1];
  if not(k in [adjm[j][l] : l in [1..degree]]) then
    print "error in path";good := false;
    end if;
  end for;
j := path[1];k := path[n];
if not(k in [adjm[j][l] : l in [1..degree]]) then
  print "error in cycle";good := false;
```

```
  end if;
// (Now check spanning)
for i := 1 to n do member[i] := false;end for;
for i := 1 to n do
  if member[path[i]] then
    print "Repeated vertices in cycle";
    good := false;
    end if;
  member[path[i]] := true;
  end for;
if good then
  // print " ";
  print "Hamilton cycle found";
  // print [path[i] : i in [1..n]];
  end if;
//
end procedure;


G:=Alt(6); n:=Order(G);  print n;
elts:=[xx : xx in G];
mt:=[[Index(elts,elts[i]*elts[j]) : i in [1..n]] : j in [1..n]];
invols:={j : j in [1..n] | Order(elts[j]) eq 2};

repeat S:=Random(Subsets(invols,3)); until sub<G|[elts[j] : j in S]> eq G;
print #S, Order(sub<G|[elts[j] : j in S]>);

edges:={{i,mt[i][j]} : i in [1..n], j in S};
cgs:=Graph<n|edges>;
print "Cayley graph of order",Order(cgs),
"valency",Degree(Rep(VertexSet(cgs))),
  "girth",Girth(cgs),"diameter",Diameter(cgs);

FindHamiltonCycle(cgs);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:01:58 on jones    [Seed = 1848425968]
Type ? for help.  Type <Ctrl>-D to quit.
>
> procedure FindHamiltonCycle(gr);
procedure> //
procedure> // *** This procedure is due to Nick Wormald (in the 1980s) ***
procedure> //
procedure> maxverts := 12000;
procedure> maxdegree := 6;
procedure> dvec:=[ 0 : i in [1..maxdegree]];
procedure> adjm := [dvec: j in [1..maxverts]];
procedure> path := [ 0 : j in [1..maxverts]];
procedure> //
procedure> newpath:=path;
procedure> member := [true: j in [1..maxverts]];
procedure> ignore:=1;degree:=3;
procedure> //
procedure> n:=Order(gr);
procedure> vts:=VertexSet(gr);
procedure> edges:={{i,j} : i in [1..n], j in [1..n] | vts!i in
Neighbours(vts!\
j)};
procedure> //
procedure> for i := 1 to n do
procedure|for>   member[i] := false;  j:=0;
procedure|for>   for k := 1 to n do if ({i,k} in edges) then j:=j+1;adjm[i]
[j]\
:=k;end if;end for;
procedure|for>   end for;
procedure> //Now checking consistency of data
procedure> print "Checking consistency of data";
```

```
procedure> good := true;
procedure> for i := 1 to n do for j := 1 to degree do
procedure|for|for>   k := adjm[i][j]; test := 1;
procedure|for|for>   while (adjm[k][test] ne i) do
procedure|for|for|while>     if (test lt degree) then test := test + 1;
else g\
ood := false;print "Data inconsistent"; end if;
procedure|for|for|while>     end while;
procedure|for|for>   end for;end for;
procedure> if good then print " ... OK";end if;
procedure> //
procedure> length := 2;
procedure> path[1] := 1;path[2] := adjm[1][1];
procedure> member[path[1]] := true;member[path[2]] := true;
procedure> done := false;
procedure> counter := 0;bigcount := 0;longest := 0; longer := false;
procedure> while good and not(done) do
procedure|while>   counter := counter + 1;
procedure|while>   if ((counter div 10) * 10 eq counter) then // reverse
path
procedure|while|if>     for i := 1 to length do newpath[i] := path[length +
1 \
- i]; end for;
procedure|while|if>     for i := 1 to length do path[i] := newpath[i]; end
for\
;
procedure|while|if>     end if;
procedure|while>   if (counter gt 600000) then
procedure|while|if>     bigcount := bigcount + 1;counter := 1;
procedure|while|if>     for i := 1 to n do member[i] := false;end for;
procedure|while|if>     length := 2;path[1] := 1;path[2] := adjm[1][1];
procedure|while|if>     member[path[1]] := true;member[path[2]] := true;
procedure|while|if>     end if;
procedure|while>   if ((counter div 1000) * 1000 eq counter) then
procedure|while|if>     for i := 1 to 5 do member[path[length + 1 - i]] :=
fal\
se;end for;
procedure|while|if>     length := length - 5;
procedure|while|if>     end if;
procedure|while> //  extend path
procedure|while> longer := false;
procedure|while> for i := 1 to degree do if not(longer) then
procedure|while|for|if>   if not(member[adjm[path[length]][i]]) then longer
:=\
 true;
procedure|while|for|if|if>     j := length + 1;path[j] :=
adjm[path[length]][i\
];
procedure|while|for|if|if>     length := j;member[path[j]] := true;
procedure|while|for|if|if>     end if;
procedure|while|for|if>   end if;end for;
procedure|while> //
procedure|while>   if (length gt longest) then longest := length; end if;
procedure|while>   if ((counter div 100) * 100 eq counter) then
procedure|while|if>     print "count =", counter, ", length =", length, ",
lon\
gest =", longest;
procedure|while|if>     end if;
procedure|while>   if not(longer) then
procedure|while|if> //    switch paths
procedure|while|if> repeat
procedure|while|if|repeat>   r := Random([1..1000000]);
procedure|while|if|repeat>   new := ((degree * (r-1)) div 1000000)+1;
procedure|while|if|repeat>   newvert := adjm[path[length], new];
procedure|while|if|repeat>  until newvert ne path[length - 1];
procedure|while|if> reversing := false;
procedure|while|if> for i := 1 to length do
procedure|while|if|for>   if (path[i] eq newvert) then reversing := true;
```

```
pivo\
t := i; end if;
procedure|while|if|for>   if reversing then
procedure|while|if|for|if>     if (i eq pivot) then newpath[i] := path[i];
procedure|while|if|for|if|if>       else newpath[i] := path[length + 1 +
pivot\
 - i];
procedure|while|if|for|if|if>       end if;
procedure|while|if|for|if>    else newpath[i] := path[i];
procedure|while|if|for|if>    end if;
procedure|while|if|for>   end for;
procedure|while|if> for i := 1 to length do path[i] := newpath[i];end for;
procedure|while|if> //
procedure|while|if>     end if;
procedure|while>   if (length eq n) then
procedure|while|if>     for i := 1 to degree do done := (path[n] eq
adjm[path[\
1]][i]); end for;
procedure|while|if>     end if;
procedure|while>   end while;
procedure> //
procedure> // Now doing a final check of cycle
procedure> //
procedure> // (First test the adjacencies)
procedure> for i := 1 to n - 1 do
procedure|for>   j := path[i];k := path[i + 1];
procedure|for>   if not(k in [adjm[j][l] : l in [1..degree]]) then
procedure|for|if>     print "error in path";good := false;
procedure|for|if>     end if;
procedure|for>   end for;
procedure> j := path[1];k := path[n];
procedure> if not(k in [adjm[j][l] : l in [1..degree]]) then
procedure|if>   print "error in cycle";good := false;
procedure|if>   end if;
procedure> // (Now check spanning)
procedure> for i := 1 to n do member[i] := false;end for;
procedure> for i := 1 to n do
procedure|for>   if member[path[i]] then
procedure|for|if>     print "Repeated vertices in cycle";
procedure|for|if>     good := false;
procedure|for|if>     end if;
procedure|for>   member[path[i]] := true;
procedure|for>   end for;
procedure> if good then
procedure|if>   // print " ";
procedure|if>   print "Hamilton cycle found";
procedure|if>   // print [path[i] : i in [1..n]];
procedure|if>   end if;
procedure> //
procedure> end procedure;
>
>
> G:=Alt(6); n:=Order(G);  print n;
360
> elts:=[xx : xx in G];
> mt:=[[Index(elts,elts[i]*elts[j]) : i in [1..n]] : j in [1..n]];
> invols:={j : j in [1..n] | Order(elts[j]) eq 2};
>
> repeat S:=Random(Subsets(invols,3)); until sub<G|[elts[j] : j in S]> eq
G;
> print #S, Order(sub<G|[elts[j] : j in S]>);
3 360
>
> edges:={{i,mt[i][j]} : i in [1..n], j in S};
> cgs:=Graph<n|edges>;
> print "Cayley graph of order",Order(cgs),
"valency",Degree(Rep(VertexSet(cgs\
))),
```

```
print>   "girth",Girth(cgs),"diameter",Diameter(cgs);
Cayley graph of order 360 valency 3 girth 6 diameter 11
>
> FindHamiltonCycle(cgs);
Checking consistency of data
 ... OK
count = 100 , length = 101 , longest = 101
count = 200 , length = 200 , longest = 200
count = 300 , length = 284 , longest = 284
count = 400 , length = 300 , longest = 300
count = 500 , length = 332 , longest = 332
count = 600 , length = 341 , longest = 341
count = 700 , length = 346 , longest = 346
count = 800 , length = 350 , longest = 350
count = 900 , length = 354 , longest = 354
count = 1000 , length = 351 , longest = 355
count = 1100 , length = 358 , longest = 358
count = 1200 , length = 358 , longest = 358
count = 1300 , length = 358 , longest = 358
count = 1400 , length = 358 , longest = 358
count = 1500 , length = 358 , longest = 358
count = 1600 , length = 358 , longest = 358
count = 1700 , length = 358 , longest = 358
count = 1800 , length = 358 , longest = 358
count = 1900 , length = 358 , longest = 358
count = 2000 , length = 354 , longest = 358
count = 2100 , length = 358 , longest = 358
count = 2200 , length = 358 , longest = 358
count = 2300 , length = 358 , longest = 358
count = 2400 , length = 360 , longest = 360
count = 2500 , length = 360 , longest = 360
count = 2600 , length = 360 , longest = 360
count = 2700 , length = 360 , longest = 360
count = 2800 , length = 360 , longest = 360
count = 2900 , length = 360 , longest = 360
count = 3000 , length = 356 , longest = 360
count = 3100 , length = 360 , longest = 360
count = 3200 , length = 360 , longest = 360
Hamilton cycle found
>
> quit;

Total time: 2.820 seconds, Total memory usage: 15.56MB


//////////////////////////////////////////
// Hamilton cycles in the Petersen graph? //
//////////////////////////////////////////

**** Magma input ****

V:= SetToSequence(Subsets({1..5},2));
E:= {{s,t}: s,t in [1..#V] | #(V[s] meet V[t]) eq 0};
P:= Graph<#V|E>;

FindHamiltonCycle(P);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:02:42 on jones    [Seed = 1680258094]
Type ? for help.  Type <Ctrl>-D to quit.
>

[Note: Need to insert procedure "FindHamiltonCycle" (from above example)
here]

> V:= SetToSequence(Subsets({1..5},2));
> E:= {{s,t}: s,t in [1..#V] | #(V[s] meet V[t]) eq 0};
```

```
> P:= Graph<#V|E>;
>
> FindHamiltonCycle(P);
Checking consistency of data
 ... OK
count = 100 , length = 10 , longest = 10
count = 200 , length = 10 , longest = 10
count = 300 , length = 10 , longest = 10
count = 400 , length = 10 , longest = 10
count = 500 , length = 10 , longest = 10
count = 600 , length = 10 , longest = 10
count = 700 , length = 10 , longest = 10
count = 800 , length = 10 , longest = 10
count = 900 , length = 10 , longest = 10
count = 1000 , length = 6 , longest = 10
count = 1100 , length = 10 , longest = 10
count = 1200 , length = 10 , longest = 10
count = 1300 , length = 10 , longest = 10
count = 1400 , length = 10 , longest = 10
count = 1500 , length = 10 , longest = 10
count = 1600 , length = 10 , longest = 10
count = 1700 , length = 10 , longest = 10
count = 1800 , length = 10 , longest = 10
count = 1900 , length = 10 , longest = 10
count = 2000 , length = 6 , longest = 10
count = 2100 , length = 10 , longest = 10
count = 2200 , length = 10 , longest = 10
count = 2300 , length = 10 , longest = 10
count = 2400 , length = 10 , longest = 10
count = 2500 , length = 10 , longest = 10
count = 2600 , length = 10 , longest = 10
count = 2700 , length = 10 , longest = 10
count = 2800 , length = 10 , longest = 10
count = 2900 , length = 10 , longest = 10
count = 3000 , length = 6 , longest = 10
count = 3100 , length = 10 , longest = 10
count = 3200 , length = 10 , longest = 10
count = 3300 , length = 10 , longest = 10
count = 3400 , length = 10 , longest = 10
count = 3500 , length = 10 , longest = 10
count = 3600 , length = 10 , longest = 10
count = 3700 , length = 10 , longest = 10
count = 3800 , length = 10 , longest = 10
count = 3900 , length = 10 , longest = 10
count = 4000 , length = 6 , longest = 10
count = 4100 , length = 10 , longest = 10
count = 4200 , length = 10 , longest = 10
count = 4300 , length = 10 , longest = 10
count = 4400 , length = 10 , longest = 10
count = 4500 , length = 10 , longest = 10
count = 4600 , length = 10 , longest = 10
count = 4700 , length = 10 , longest = 10
count = 4800 , length = 10 , longest = 10
count = 4900 , length = 10 , longest = 10
count = 5000 , length = 6 , longest = 10
count = 5100 , length = 10 , longest = 10
count = 5200 , length = 10 , longest = 10
count = 5300 , length = 10 , longest = 10
count = 5400 , length = 10 , longest = 10
count = 5500 , length = 10 , longest = 10
count = 5600 , length = 10 , longest = 10
count = 5700 , length = 10 , longest = 10
count = 5800 , length = 10 , longest = 10
count = 5900 , length = 10 , longest = 10
count = 6000 , length = 6 , longest = 10
count = 6100 , length = 10 , longest = 10
count = 6200 , length = 10 , longest = 10
count = 6300 , length = 10 , longest = 10
```

```
count = 6400 , length = 10 , longest = 10
count = 6500 , length = 10 , longest = 10
count = 6600 , length = 10 , longest = 10
count = 6700 , length = 10 , longest = 10
count = 6800 , length = 10 , longest = 10
count = 6900 , length = 10 , longest = 10
count = 7000 , length = 6 , longest = 10
count = 7100 , length = 10 , longest = 10
count = 7200 , length = 10 , longest = 10
count = 7300 , length = 10 , longest = 10
count = 7400 , length = 10 , longest = 10
count = 7500 , length = 10 , longest = 10
count = 7600 , length = 10 , longest = 10
count = 7700 , length = 10 , longest = 10
count = 7800 , length = 10 , longest = 10
count = 7900 , length = 10 , longest = 10
count = 8000 , length = 6 , longest = 10
count = 8100 , length = 10 , longest = 10
count = 8200 , length = 10 , longest = 10
count = 8300 , length = 10 , longest = 10
count = 8400 , length = 10 , longest = 10
count = 8500 , length = 10 , longest = 10
count = 8600 , length = 10 , longest = 10
count = 8700 , length = 10 , longest = 10
count = 8800 , length = 10 , longest = 10
count = 8900 , length = 10 , longest = 10
count = 9000 , length = 6 , longest = 10
count = 9100 , length = 10 , longest = 10
count = 9200 , length = 10 , longest = 10
count = 9300 , length = 10 , longest = 10
count = 9400 , length = 10 , longest = 10
count = 9500 , length = 10 , longest = 10
count = 9600 , length = 10 , longest = 10
count = 9700 , length = 10 , longest = 10
count = 9800 , length = 10 , longest = 10
count = 9900 , length = 10 , longest = 10
count = 10000 , length = 6 , longest = 10
count = 10100 , length = 10 , longest = 10
count = 10200 , length = 10 , longest = 10
count = 10300 , length = 10 , longest = 10
count = 10400 , length = 10 , longest = 10
count = 10500 , length = 10 , longest = 10
count = 10600 , length = 10 , longest = 10
count = 10700 , length = 10 , longest = 10
count = 10800 , length = 10 , longest = 10
count = 10900 , length = 10 , longest = 10
count = 11000 , length = 6 , longest = 10
count = 11100 , length = 10 , longest = 10
count = 11200 , length = 10 , longest = 10
count = 11300 , length = 10 , longest = 10
count = 11400 , length = 10 , longest = 10
count = 11500 , length = 10 , longest = 10
count = 11600 , length = 10 , longest = 10
count = 11700 , length = 10 , longest = 10
count = 11800 , length = 10 , longest = 10
count = 11900 , length = 10 , longest = 10
count = 12000 , length = 6 , longest = 10
count = 12100 , length = 10 , longest = 10
count = 12200 , length = 10 , longest = 10
count = 12300 , length = 10 , longest = 10
count = 12400 , length = 10 , longest = 10
count = 12500 , length = 10 , longest = 10
count = 12600 , length = 10 , longest = 10
count = 12700 , length = 10 , longest = 10
count = 12800 , length = 10 , longest = 10
count = 12900 , length = 10 , longest = 10
count = 13000 , length = 6 , longest = 10
count = 13100 , length = 10 , longest = 10
```

```
count = 13200 , length = 10 , longest = 10
count = 13300 , length = 10 , longest = 10
count = 13400 , length = 10 , longest = 10
count = 13500 , length = 10 , longest = 10
count = 13600 , length = 10 , longest = 10
count = 13700 , length = 10 , longest = 10
count = 13800 , length = 10 , longest = 10
count = 13900 , length = 10 , longest = 10
count = 14000 , length = 6 , longest = 10
count = 14100 , length = 10 , longest = 10
count = 14200 , length = 10 , longest = 10
count = 14300 , length = 10 , longest = 10
count = 14400 , length = 10 , longest = 10
count = 14500 , length = 10 , longest = 10
count = 14600 , length = 10 , longest = 10
count = 14700 , length = 10 , longest = 10
count = 14800 , length = 10 , longest = 10
count = 14900 , length = 10 , longest = 10
count = 15000 , length = 6 , longest = 10
count = 15100 , length = 10 , longest = 10
count = 15200 , length = 10 , longest = 10
count = 15300 , length = 10 , longest = 10
count = 15400 , length = 10 , longest = 10
count = 15500 , length = 10 , longest = 10
count = 15600 , length = 10 , longest = 10
count = 15700 , length = 10 , longest = 10
count = 15800 , length = 10 , longest = 10
count = 15900 , length = 10 , longest = 10
count = 16000 , length = 6 , longest = 10
count = 16100 , length = 10 , longest = 10
count = 16200 , length = 10 , longest = 10
count = 16300 , length = 10 , longest = 10
count = 16400 , length = 10 , longest = 10
count = 16500 , length = 10 , longest = 10
count = 16600 , length = 10 , longest = 10
count = 16700 , length = 10 , longest = 10
count = 16800 , length = 10 , longest = 10
count = 16900 , length = 10 , longest = 10
count = 17000 , length = 6 , longest = 10
count = 17100 , length = 10 , longest = 10
count = 17200 , length = 10 , longest = 10
count = 17300 , length = 10 , longest = 10
count = 17400 , length = 10 , longest = 10
count = 17500 , length = 10 , longest = 10
count = 17600 , length = 10 , longest = 10
count = 17700 , length = 10 , longest = 10
count = 17800 , length = 10 , longest = 10
count = 17900 , length = 10 , longest = 10
count = 18000 , length = 6 , longest = 10
count = 18100 , length = 10 , longest = 10
count = 18200 , length = 10 , longest = 10
count = 18300 , length = 10 , longest = 10
count = 18400 , length = 10 , longest = 10
count = 18500 , length = 10 , longest = 10
count = 18600 , length = 10 , longest = 10
count = 18700 , length = 10 , longest = 10
count = 18800 , length = 10 , longest = 10
count = 18900 , length = 10 , longest = 10
count = 19000 , length = 6 , longest = 10
count = 19100 , length = 10 , longest = 10
count = 19200 , length = 10 , longest = 10
count = 19300 , length = 10 , longest = 10
count = 19400 , length = 10 , longest = 10
count = 19500 , length = 10 , longest = 10
count = 19600 , length = 10 , longest = 10
count = 19700 , length = 10 , longest = 10
count = 19800 , length = 10 , longest = 10
count = 19900 , length = 10 , longest = 10
```

```
count = 20000 , length = 6 , longest = 10
count = 20100 , length = 10 , longest = 10
count = 20200 , length = 10 , longest = 10
count = 20300 , length = 10 , longest = 10
count = 20400 , length = 10 , longest = 10
count = 20500 , length = 10 , longest = 10
count = 20600 , length = 10 , longest = 10
count = 20700 , length = 10 , longest = 10
count = 20800 , length = 10 , longest = 10
count = 20900 , length = 10 , longest = 10
count = 21000 , length = 6 , longest = 10
count = 21100 , length = 10 , longest = 10
count = 21200 , length = 10 , longest = 10
count = 21300 , length = 10 , longest = 10
count = 21400 , length = 10 , longest = 10
count = 21500 , length = 10 , longest = 10
count = 21600 , length = 10 , longest = 10
count = 21700 , length = 10 , longest = 10
count = 21800 , length = 10 , longest = 10
count = 21900 , length = 10 , longest = 10
count = 22000 , length = 6 , longest = 10
count = 22100 , length = 10 , longest = 10
count = 22200 , length = 10 , longest = 10
count = 22300 , length = 10 , longest = 10
count = 22400 , length = 10 , longest = 10
count = 22500 , length = 10 , longest = 10
count = 22600 , length = 10 , longest = 10
count = 22700 , length = 10 , longest = 10
count = 22800 , length = 10 , longest = 10
count = 22900 , length = 10 , longest = 10
count = 23000 , length = 6 , longest = 10
count = 23100 , length = 10 , longest = 10
count = 23200 , length = 10 , longest = 10
count = 23300 , length = 10 , longest = 10
count = 23400 , length = 10 , longest = 10
count = 23500 , length = 10 , longest = 10
count = 23600 , length = 10 , longest = 10
count = 23700 , length = 10 , longest = 10
count = 23800 , length = 10 , longest = 10
count = 23900 , length = 10 , longest = 10
count = 24000 , length = 6 , longest = 10
count = 24100 , length = 10 , longest = 10
count = 24200 , length = 10 , longest = 10
count = 24300 , length = 10 , longest = 10
count = 24400 , length = 10 , longest = 10
count = 24500 , length = 10 , longest = 10
count = 24600 , length = 10 , longest = 10
count = 24700 , length = 10 , longest = 10
count = 24800 , length = 10 , longest = 10
count = 24900 , length = 10 , longest = 10
count = 25000 , length = 6 , longest = 10
count = 25100 , length = 10 , longest = 10
count = 25200 , length = 10 , longest = 10
count = 25300 , length = 10 , longest = 10
count = 25400 , length = 10 , longest = 10
count = 25500 , length = 10 , longest = 10
count = 25600 , length = 10 , longest = 10
count = 25700 , length = 10 , longest = 10
count = 25800 , length = 10 , longest = 10
count = 25900 , length = 10 , longest = 10
count = 26000 , length = 6 , longest = 10
count = 26100 , length = 10 , longest = 10
count = 26200 , length = 10 , longest = 10
count = 26300 , length = 10 , longest = 10
count = 26400 , length = 10 , longest = 10
count = 26500 , length = 10 , longest = 10
count = 26600 , length = 10 , longest = 10
count = 26700 , length = 10 , longest = 10
```

```
count = 26800 , length = 10 , longest = 10
count = 26900 , length = 10 , longest = 10
count = 27000 , length = 6 , longest = 10
count = 27100 , length = 10 , longest = 10
count = 27200 , length = 10 , longest = 10
count = 27300 , length = 10 , longest = 10
count = 27400 , length = 10 , longest = 10
count = 27500 , length = 10 , longest = 10
count = 27600 , length = 10 , longest = 10
count = 27700 , length = 10 , longest = 10
count = 27800 , length = 10 , longest = 10
count = 27900 , length = 10 , longest = 10
count = 28000 , length = 6 , longest = 10
count = 28100 , length = 10 , longest = 10
count = 28200 , length = 10 , longest = 10
count = 28300 , length = 10 , longest = 10
count = 28400 , length = 10 , longest = 10
count = 28500 , length = 10 , longest = 10
count = 28600 , length = 10 , longest = 10
count = 28700 , length = 10 , longest = 10
count = 28800 , length = 10 , longest = 10
count = 28900 , length = 10 , longest = 10
count = 29000 , length = 6 , longest = 10
count = 29100 , length = 10 , longest = 10
count = 29200 , length = 10 , longest = 10
count = 29300 , length = 10 , longest = 10
count = 29400 , length = 10 , longest = 10
count = 29500 , length = 10 , longest = 10
count = 29600 , length = 10 , longest = 10
count = 29700 , length = 10 , longest = 10
count = 29800 , length = 10 , longest = 10
count = 29900 , length = 10 , longest = 10
count = 30000 , length = 6 , longest = 10
count = 30100 , length = 10 , longest = 10
count = 30200 , length = 10 , longest = 10
count = 30300 , length = 10 , longest = 10
count = 30400 , length = 10 , longest = 10
count = 30500 , length = 10 , longest = 10
count = 30600 , length = 10 , longest = 10
count = 30700 , length = 10 , longest = 10
count = 30800 , length = 10 , longest = 10
count = 30900 , length = 10 , longest = 10
count = 31000 , length = 6 , longest = 10
count = 31100 , length = 10 , longest = 10
count = 31200 , length = 10 , longest = 10
count = 31300 , length = 10 , longest = 10
count = 31400 , length = 10 , longest = 10
count = 31500 , length = 10 , longest = 10
count = 31600 , length = 10 , longest = 10
count = 31700 , length = 10 , longest = 10
count = 31800 , length = 10 , longest = 10
count = 31900 , length = 10 , longest = 10
count = 32000 , length = 6 , longest = 10
count = 32100 , length = 10 , longest = 10
count = 32200 , length = 10 , longest = 10
count = 32300 , length = 10 , longest = 10
count = 32400 , length = 10 , longest = 10
count = 32500 , length = 10 , longest = 10
count = 32600 , length = 10 , longest = 10
count = 32700 , length = 10 , longest = 10
count = 32800 , length = 10 , longest = 10
count = 32900 , length = 10 , longest = 10
count = 33000 , length = 6 , longest = 10
count = 33100 , length = 10 , longest = 10
count = 33200 , length = 10 , longest = 10
count = 33300 , length = 10 , longest = 10
count = 33400 , length = 10 , longest = 10
count = 33500 , length = 10 , longest = 10
```

```
count = 33600 , length = 10 , longest = 10
count = 33700 , length = 10 , longest = 10
count = 33800 , length = 10 , longest = 10
count = 33900 , length = 10 , longest = 10
count = 34000 , length = 6 , longest = 10
count = 34100 , length = 10 , longest = 10
count = 34200 , length = 10 , longest = 10
count = 34300 , length = 10 , longest = 10
count = 34400 , length = 10 , longest = 10
count = 34500 , length = 10 , longest = 10
count = 34600 , length = 10 , longest = 10
count = 34700 , length = 10 , longest = 10
count = 34800 , length = 10 , longest = 10
count = 34900 , length = 10 , longest = 10
count = 35000 , length = 6 , longest = 10
count = 35100 , length = 10 , longest = 10
count = 35200 , length = 10 , longest = 10
count = 35300 , length = 10 , longest = 10
count = 35400 , length = 10 , longest = 10
count = 35500 , length = 10 , longest = 10
count = 35600 , length = 10 , longest = 10
count = 35700 , length = 10 , longest = 10
count = 35800 , length = 10 , longest = 10
count = 35900 , length = 10 , longest = 10
count = 36000 , length = 6 , longest = 10
count = 36100 , length = 10 , longest = 10
count = 36200 , length = 10 , longest = 10
count = 36300 , length = 10 , longest = 10
count = 36400 , length = 10 , longest = 10
count = 36500 , length = 10 , longest = 10
count = 36600 , length = 10 , longest = 10
count = 36700 , length = 10 , longest = 10
count = 36800 , length = 10 , longest = 10
count = 36900 , length = 10 , longest = 10
count = 37000 , length = 6 , longest = 10
count = 37100 , length = 10 , longest = 10
count = 37200 , length = 10 , longest = 10
count = 37300 , length = 10 , longest = 10
count = 37400 , length = 10 , longest = 10
count = 37500 , length = 10 , longest = 10
count = 37600 , length = 10 , longest = 10
count = 37700 , length = 10 , longest = 10
count = 37800 , length = 10 , longest = 10
count = 37900 , length = 10 , longest = 10
count = 38000 , length = 6 , longest = 10
count = 38100 , length = 10 , longest = 10
count = 38200 , length = 10 , longest = 10
count = 38300 , length = 10 , longest = 10
count = 38400 , length = 10 , longest = 10
count = 38500 , length = 10 , longest = 10
count = 38600 , length = 10 , longest = 10
count = 38700 , length = 10 , longest = 10
count = 38800 , length = 10 , longest = 10
count = 38900 , length = 10 , longest = 10
count = 39000 , length = 6 , longest = 10
count = 39100 , length = 10 , longest = 10
count = 39200 , length = 10 , longest = 10
count = 39300 , length = 10 , longest = 10
count = 39400 , length = 10 , longest = 10
count = 39500 , length = 10 , longest = 10
count = 39600 , length = 10 , longest = 10
count = 39700 , length = 10 , longest = 10
count = 39800 , length = 10 , longest = 10
count = 39900 , length = 10 , longest = 10
count = 40000 , length = 6 , longest = 10
count = 40100 , length = 10 , longest = 10
count = 40200 , length = 10 , longest = 10
count = 40300 , length = 10 , longest = 10
```

```
count = 40400 , length = 10 , longest = 10
count = 40500 , length = 10 , longest = 10
count = 40600 , length = 10 , longest = 10
count = 40700 , length = 10 , longest = 10
count = 40800 , length = 10 , longest = 10
count = 40900 , length = 10 , longest = 10
count = 41000 , length = 6 , longest = 10
count = 41100 , length = 10 , longest = 10
count = 41200 , length = 10 , longest = 10
count = 41300 , length = 10 , longest = 10
count = 41400 , length = 10 , longest = 10
count = 41500 , length = 10 , longest = 10
count = 41600 , length = 10 , longest = 10
count = 41700 , length = 10 , longest = 10
count = 41800 , length = 10 , longest = 10
count = 41900 , length = 10 , longest = 10
count = 42000 , length = 6 , longest = 10
count = 42100 , length = 10 , longest = 10
count = 42200 , length = 10 , longest = 10
count = 42300 , length = 10 , longest = 10
count = 42400 , length = 10 , longest = 10
count = 42500 , length = 10 , longest = 10
count = 42600 , length = 10 , longest = 10
count = 42700 , length = 10 , longest = 10
count = 42800 , length = 10 , longest = 10
count = 42900 , length = 10 , longest = 10
count = 43000 , length = 6 , longest = 10
count = 43100 , length = 10 , longest = 10
count = 43200 , length = 10 , longest = 10
count = 43300 , length = 10 , longest = 10
count = 43400 , length = 10 , longest = 10
count = 43500 , length = 10 , longest = 10
count = 43600 , length = 10 , longest = 10
count = 43700 , length = 10 , longest = 10
count = 43800 , length = 10 , longest = 10
count = 43900 , length = 10 , longest = 10
count = 44000 , length = 6 , longest = 10
count = 44100 , length = 10 , longest = 10
count = 44200 , length = 10 , longest = 10
count = 44300 , length = 10 , longest = 10
count = 44400 , length = 10 , longest = 10
count = 44500 , length = 10 , longest = 10
count = 44600 , length = 10 , longest = 10
count = 44700 , length = 10 , longest = 10
count = 44800 , length = 10 , longest = 10
count = 44900 , length = 10 , longest = 10
count = 45000 , length = 6 , longest = 10
count = 45100 , length = 10 , longest = 10
count = 45200 , length = 10 , longest = 10
count = 45300 , length = 10 , longest = 10
count = 45400 , length = 10 , longest = 10
count = 45500 , length = 10 , longest = 10
count = 45600 , length = 10 , longest = 10
count = 45700 , length = 10 , longest = 10
count = 45800 , length = 10 , longest = 10
count = 45900 , length = 10 , longest = 10
count = 46000 , length = 6 , longest = 10
count = 46100 , length = 10 , longest = 10
count = 46200 , length = 10 , longest = 10
count = 46300 , length = 10 , longest = 10
count = 46400 , length = 10 , longest = 10
count = 46500 , length = 10 , longest = 10
count = 46600 , length = 10 , longest = 10
count = 46700 , length = 10 , longest = 10
count = 46800 , length = 10 , longest = 10
count = 46900 , length = 10 , longest = 10
count = 47000 , length = 6 , longest = 10
count = 47100 , length = 10 , longest = 10
```

```
count = 47200 , length = 10 , longest = 10
count = 47300 , length = 10 , longest = 10
count = 47400 , length = 10 , longest = 10
count = 47500 , length = 10 , longest = 10
count = 47600 , length = 10 , longest = 10
count = 47700 , length = 10 , longest = 10
count = 47800 , length = 10 , longest = 10
count = 47900 , length = 10 , longest = 10
count = 48000 , length = 6 , longest = 10
count = 48100 , length = 10 , longest = 10
count = 48200 , length = 10 , longest = 10
count = 48300 , length = 10 , longest = 10
count = 48400 , length = 10 , longest = 10
count = 48500 , length = 10 , longest = 10
count = 48600 , length = 10 , longest = 10
count = 48700 , length = 10 , longest = 10
count = 48800 , length = 10 , longest = 10
count = 48900 , length = 10 , longest = 10
count = 49000 , length = 6 , longest = 10
count = 49100 , length = 10 , longest = 10
count = 49200 , length = 10 , longest = 10
count = 49300 , length = 10 , longest = 10
count = 49400 , length = 10 , longest = 10
count = 49500 , length = 10 , longest = 10
count = 49600 , length = 10 , longest = 10
count = 49700 , length = 10 , longest = 10
count = 49800 , length = 10 , longest = 10
count = 49900 , length = 10 , longest = 10
count = 50000 , length = 6 , longest = 10
count = 50100 , length = 10 , longest = 10
count = 50200 , length = 10 , longest = 10
count = 50300 , length = 10 , longest = 10
count = 50400 , length = 10 , longest = 10
count = 50500 , length = 10 , longest = 10
count = 50600 , length = 10 , longest = 10
count = 50700 , length = 10 , longest = 10
count = 50800 , length = 10 , longest = 10
count = 50900 , length = 10 , longest = 10
count = 51000 , length = 6 , longest = 10
count = 51100 , length = 10 , longest = 10
count = 51200 , length = 10 , longest = 10
count = 51300 , length = 10 , longest = 10
count = 51400 , length = 10 , longest = 10
count = 51500 , length = 10 , longest = 10
count = 51600 , length = 10 , longest = 10
count = 51700 , length = 10 , longest = 10
count = 51800 , length = 10 , longest = 10
count = 51900 , length = 10 , longest = 10
count = 52000 , length = 6 , longest = 10
count = 52100 , length = 10 , longest = 10
count = 52200 , length = 10 , longest = 10
count = 52300 , length = 10 , longest = 10
count = 52400 , length = 10 , longest = 10
count = 52500 , length = 10 , longest = 10
count = 52600 , length = 10 , longest = 10
count = 52700 , length = 10 , longest = 10
count = 52800 , length = 10 , longest = 10
count = 52900 , length = 10 , longest = 10
count = 53000 , length = 6 , longest = 10
count = 53100 , length = 10 , longest = 10
count = 53200 , length = 10 , longest = 10
count = 53300 , length = 10 , longest = 10
count = 53400 , length = 10 , longest = 10
count = 53500 , length = 10 , longest = 10
count = 53600 , length = 10 , longest = 10
count = 53700 , length = 10 , longest = 10
count = 53800 , length = 10 , longest = 10
count = 53900 , length = 10 , longest = 10
```

```
count = 54000 , length = 6 , longest = 10
count = 54100 , length = 10 , longest = 10
count = 54200 , length = 10 , longest = 10
count = 54300 , length = 10 , longest = 10
count = 54400 , length = 10 , longest = 10
count = 54500 , length = 10 , longest = 10
count = 54600 , length = 10 , longest = 10
count = 54700 , length = 10 , longest = 10
count = 54800 , length = 10 , longest = 10
count = 54900 , length = 10 , longest = 10
count = 55000 , length = 6 , longest = 10
count = 55100 , length = 10 , longest = 10
count = 55200 , length = 10 , longest = 10
count = 55300 , length = 10 , longest = 10
count = 55400 , length = 10 , longest = 10
count = 55500 , length = 10 , longest = 10
count = 55600 , length = 10 , longest = 10
count = 55700 , length = 10 , longest = 10
count = 55800 , length = 10 , longest = 10
count = 55900 , length = 10 , longest = 10
count = 56000 , length = 6 , longest = 10
count = 56100 , length = 10 , longest = 10
count = 56200 , length = 10 , longest = 10
count = 56300 , length = 10 , longest = 10
count = 56400 , length = 10 , longest = 10
count = 56500 , length = 10 , longest = 10
count = 56600 , length = 10 , longest = 10
count = 56700 , length = 10 , longest = 10
count = 56800 , length = 10 , longest = 10
count = 56900 , length = 10 , longest = 10
count = 57000 , length = 6 , longest = 10
count = 57100 , length = 10 , longest = 10
count = 57200 , length = 10 , longest = 10
count = 57300 , length = 10 , longest = 10
count = 57400 , length = 10 , longest = 10
count = 57500 , length = 10 , longest = 10
count = 57600 , length = 10 , longest = 10
count = 57700 , length = 10 , longest = 10
count = 57800 , length = 10 , longest = 10
count = 57900 , length = 10 , longest = 10
count = 58000 , length = 6 , longest = 10
count = 58100 , length = 10 , longest = 10
count = 58200 , length = 10 , longest = 10
count = 58300 , length = 10 , longest = 10
count = 58400 , length = 10 , longest = 10
count = 58500 , length = 10 , longest = 10
count = 58600 , length = 10 , longest = 10
count = 58700 , length = 10 , longest = 10
count = 58800 , length = 10 , longest = 10
count = 58900 , length = 10 , longest = 10
count = 59000 , length = 6 , longest = 10
count = 59100 , length = 10 , longest = 10
count = 59200 , length = 10 , longest = 10
count = 59300 , length = 10 , longest = 10
count = 59400 , length = 10 , longest = 10
count = 59500 , length = 10 , longest = 10
count = 59600 , length = 10 , longest = 10
count = 59700 , length = 10 , longest = 10
count = 59800 , length = 10 , longest = 10
count = 59900 , length = 10 , longest = 10
count = 60000 , length = 6 , longest = 10
count = 60100 , length = 10 , longest = 10
count = 60200 , length = 10 , longest = 10
count = 60300 , length = 10 , longest = 10
count = 60400 , length = 10 , longest = 10
count = 60500 , length = 10 , longest = 10
count = 60600 , length = 10 , longest = 10
count = 60700 , length = 10 , longest = 10
```

```
count = 60800 , length = 10 , longest = 10
count = 60900 , length = 10 , longest = 10
count = 61000 , length = 6 , longest = 10
count = 61100 , length = 10 , longest = 10
count = 61200 , length = 10 , longest = 10
count = 61300 , length = 10 , longest = 10
count = 61400 , length = 10 , longest = 10
count = 61500 , length = 10 , longest = 10
count = 61600 , length = 10 , longest = 10
count = 61700 , length = 10 , longest = 10
count = 61800 , length = 10 , longest = 10
count = 61900 , length = 10 , longest = 10
count = 62000 , length = 6 , longest = 10
count = 62100 , length = 10 , longest = 10
count = 62200 , length = 10 , longest = 10
count = 62300 , length = 10 , longest = 10
count = 62400 , length = 10 , longest = 10
count = 62500 , length = 10 , longest = 10
count = 62600 , length = 10 , longest = 10
count = 62700 , length = 10 , longest = 10
count = 62800 , length = 10 , longest = 10
count = 62900 , length = 10 , longest = 10
count = 63000 , length = 6 , longest = 10
count = 63100 , length = 10 , longest = 10
count = 63200 , length = 10 , longest = 10
count = 63300 , length = 10 , longest = 10
count = 63400 , length = 10 , longest = 10
count = 63500 , length = 10 , longest = 10
count = 63600 , length = 10 , longest = 10
count = 63700 , length = 10 , longest = 10
count = 63800 , length = 10 , longest = 10
count = 63900 , length = 10 , longest = 10
^C
[Interrupted]
>
> quit;

Total time: 3.150 seconds, Total memory usage: 9.53MB



////////////////////////////////////////////////////////////////////////
// Lecture 2 ///////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

.....................................................................
Demonstration 4:  Use of MAGMA to study permutation groups
.....................................................................

//////////////////////////
// Dihedral group D_5  //
//////////////////////////

**** Magma input ****

D:=DihedralGroup(5);
print D;

print Orbits(D); IsTransitive(D);

print AllPartitions(D); IsPrimitive(D);

IsAlternating(D);IsSymmetric(D);

print Base(D);  BasicOrbitLengths(D);

cc:=ConjugacyClasses(D); print cc;
```

```
for i in [1..#cc] do g:=cc[i][3];
  print i," Order",cc[i][1]," Class size",cc[i][2],
    " Semiregular?", IsSemiregular(sub<D|g>),
    " CycStr",CycleStructure(g);
  end for;

print CompositionFactors(D);

print NormalSubgroups(D);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:06:28 on jones    [Seed = 1949738264]
Type ? for help.  Type <Ctrl>-D to quit.
>
> D:=DihedralGroup(5);
> print D;
Permutation group D acting on a set of cardinality 5
Order = 10 = 2 * 5
    (1, 2, 3, 4, 5)
    (1, 5)(2, 4)
>
> print Orbits(D); IsTransitive(D);
[
    GSet{@ 1, 2, 5, 3, 4 @}
]
true
>
> print AllPartitions(D); IsPrimitive(D);
{}
true
>
> IsAlternating(D);IsSymmetric(D);
false
false
>
> print Base(D);  BasicOrbitLengths(D);
[ 3, 1 ]
[ 5, 2 ]
>
> cc:=ConjugacyClasses(D); print cc;
Conjugacy Classes of group D
----------------------------
[1]     Order 1       Length 1
        Rep Id(D)

[2]     Order 2       Length 5
        Rep (1, 5)(2, 4)

[3]     Order 5       Length 2
        Rep (1, 2, 3, 4, 5)

[4]     Order 5       Length 2
        Rep (1, 3, 5, 2, 4)


>
> for i in [1..#cc] do g:=cc[i][3];
for>   print i," Order",cc[i][1]," Class size",cc[i][2],
for|print>     " Semiregular?", IsSemiregular(sub<D|g>),
for|print>     " CycStr",CycleStructure(g);
for>   end for;
1  Order 1  Class size 1  Semiregular? true  CycStr [ <1, 5> ]
2  Order 2  Class size 5  Semiregular? false  CycStr [ <2, 2>, <1, 1> ]
3  Order 5  Class size 2  Semiregular? true  CycStr [ <5, 1> ]
4  Order 5  Class size 2  Semiregular? true  CycStr [ <5, 1> ]
>
> print CompositionFactors(D);
```

```
    G
    |  Cyclic(2)
    *
    |  Cyclic(5)
    1
>
> print NormalSubgroups(D);
Conjugacy classes of subgroups
------------------------------

[1]     Order 1            Length 1
        Permutation group acting on a set of cardinality 5
        Order = 1
[2]     Order 5            Length 1
        Permutation group acting on a set of cardinality 5
        Order = 5
            (1, 4, 2, 5, 3)
[3]     Order 10           Length 1
        Permutation group acting on a set of cardinality 5
        Order = 10 = 2 * 5
            (1, 5)(2, 4)
            (1, 4, 2, 5, 3)
>
> quit;

Total time: 0.430 seconds, Total memory usage: 11.03MB


//////////////////////////
// Dihedral group D_6  //
//////////////////////////

**** Magma input ****

D:=DihedralGroup(6);
print D;

print Orbits(D); IsTransitive(D);
print AllPartitions(D); IsPrimitive(D);
IsAlternating(D);IsSymmetric(D);

print Base(D);  BasicOrbitLengths(D);

cc:=ConjugacyClasses(D);
for i in [1..#cc] do g:=cc[i][3];
  print i," Order",cc[i][1]," Class size",cc[i][2],
    " Semiregular?", IsSemiregular(sub<D|g>),
    " CycStr",CycleStructure(g);
  end for;

print CompositionFactors(D);

print NormalSubgroups(D);

print MaximalSubgroups(D);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:07:05 on jones    [Seed = 1309449999]
Type ? for help.  Type <Ctrl>-D to quit.
>
> D:=DihedralGroup(6);
> print D;
Permutation group D acting on a set of cardinality 6
Order = 12 = 2^2 * 3
    (1, 2, 3, 4, 5, 6)
    (1, 6)(2, 5)(3, 4)
>
```

```
> print Orbits(D); IsTransitive(D);
[
    GSet{@ 1, 2, 6, 3, 5, 4 @}
]
true
> print AllPartitions(D); IsPrimitive(D);
{
    { 1, 3, 5 },
    { 1, 4 }
}
false
> IsAlternating(D);IsSymmetric(D);
false
false
>
> print Base(D);  BasicOrbitLengths(D);
[ 1, 2 ]
[ 6, 2 ]
>
> cc:=ConjugacyClasses(D);
> for i in [1..#cc] do g:=cc[i][3];
for>   print i," Order",cc[i][1]," Class size",cc[i][2],
for|print>     " Semiregular?", IsSemiregular(sub<D|g>),
for|print>     " CycStr",CycleStructure(g);
for>   end for;
1  Order 1  Class size 1  Semiregular? true  CycStr [ <1, 6> ]
2  Order 2  Class size 1  Semiregular? true  CycStr [ <2, 3> ]
3  Order 2  Class size 3  Semiregular? true  CycStr [ <2, 3> ]
4  Order 2  Class size 3  Semiregular? false  CycStr [ <2, 2>, <1, 2> ]
5  Order 3  Class size 2  Semiregular? true  CycStr [ <3, 2> ]
6  Order 6  Class size 2  Semiregular? true  CycStr [ <6, 1> ]
>
> print CompositionFactors(D);
    G
    |  Cyclic(2)
    *
    |  Cyclic(2)
    *
    |  Cyclic(3)
    1
>
> print NormalSubgroups(D);
Conjugacy classes of subgroups
------------------------------

[1]     Order 1            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 1
[2]     Order 2            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 2
            (1, 4)(2, 5)(3, 6)
[3]     Order 3            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 3
            (1, 5, 3)(2, 6, 4)
[4]     Order 6            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 6 = 2 * 3
            (2, 6)(3, 5)
            (1, 5, 3)(2, 6, 4)
[5]     Order 6            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 6 = 2 * 3
            (1, 2, 3, 4, 5, 6)
            (1, 5, 3)(2, 6, 4)
[6]     Order 6            Length 1
        Permutation group acting on a set of cardinality 6
```

```
        Order = 6 = 2 * 3
            (1, 2)(3, 6)(4, 5)
            (1, 5, 3)(2, 6, 4)
[7]     Order 12            Length 1
        Permutation group acting on a set of cardinality 6
        Order = 12 = 2^2 * 3
            (2, 6)(3, 5)
            (1, 2, 3, 4, 5, 6)
            (1, 5, 3)(2, 6, 4)
>
> print MaximalSubgroups(D);
Conjugacy classes of subgroups
------------------------------

[1]     Order 4             Length 3
        Permutation group acting on a set of cardinality 6
        Order = 4 = 2^2
            (2, 6)(3, 5)
            (1, 4)(2, 5)(3, 6)
[2]     Order 6             Length 1
        Permutation group acting on a set of cardinality 6
        Order = 6 = 2 * 3
            (1, 2)(3, 6)(4, 5)
            (1, 5, 3)(2, 6, 4)
[3]     Order 6             Length 1
        Permutation group acting on a set of cardinality 6
        Order = 6 = 2 * 3
            (1, 2, 3, 4, 5, 6)
            (1, 5, 3)(2, 6, 4)
[4]     Order 6             Length 1
        Permutation group acting on a set of cardinality 6
        Order = 6 = 2 * 3
            (2, 6)(3, 5)
            (1, 5, 3)(2, 6, 4)
>
> quit;

Total time: 0.500 seconds, Total memory usage: 11.03MB


//////////////////////////
// Symmetric group S_5  //
//////////////////////////

**** Magma input ****

S:=SymmetricGroup(5);
print S;

print IsTransitive(S),IsPrimitive(S);
IsAlternating(S);IsSymmetric(S);

print Base(S);  BasicOrbitLengths(S);

cc:=ConjugacyClasses(S);
for i in [1..#cc] do g:=cc[i][3];
  print i," Order",cc[i][1]," Class size",cc[i][2],
    " Semiregular?", IsSemiregular(sub<S|g>),
    " CycStr",CycleStructure(g);
  end for;

print CharacterTable(S);

print CompositionFactors(S);

ns:=NormalSubgroups(S); print [Order(ns[i]`subgroup) : i in [1..#ns]];

ms:=MaximalSubgroups(S); print [Order(ms[i]`subgroup) : i in [1..#ms]];
```

```
**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:07:35 on jones    [Seed = 1107333997]
Type ? for help.  Type <Ctrl>-D to quit.
>
> S:=SymmetricGroup(5);
> print S;
Symmetric group S acting on a set of cardinality 5
Order = 120 = 2^3 * 3 * 5
>
> print IsTransitive(S),IsPrimitive(S);
true true
> IsAlternating(S);IsSymmetric(S);
false
true
>
> print Base(S);  BasicOrbitLengths(S);
[ 1, 2, 3, 4 ]
[ 5, 4, 3, 2 ]
>
> cc:=ConjugacyClasses(S);
> for i in [1..#cc] do g:=cc[i][3];
for>   print i," Order",cc[i][1]," Class size",cc[i][2],
for|print>     " Semiregular?", IsSemiregular(sub<S|g>),
for|print>     " CycStr",CycleStructure(g);
for>   end for;
1  Order 1  Class size 1  Semiregular? true  CycStr [ <1, 5> ]
2  Order 2  Class size 10  Semiregular? false  CycStr [ <2, 1>, <1, 3> ]
3  Order 2  Class size 15  Semiregular? false  CycStr [ <2, 2>, <1, 1> ]
4  Order 3  Class size 20  Semiregular? false  CycStr [ <3, 1>, <1, 2> ]
5  Order 4  Class size 30  Semiregular? false  CycStr [ <4, 1>, <1, 1> ]
6  Order 5  Class size 24  Semiregular? true  CycStr [ <5, 1> ]
7  Order 6  Class size 20  Semiregular? false  CycStr [ <3, 1>, <2, 1> ]
>
> print CharacterTable(S);


Character Table of Group S
--------------------------


-----------------------------
Class |   1  2  3  4  5  6  7
Size  |   1 10 15 20 30 24 20
Order |   1  2  2  3  4  5  6
-----------------------------
p  =  2   1  1  1  4  3  6  4
p  =  3   1  2  3  1  5  6  2
p  =  5   1  2  3  4  5  1  7
-----------------------------
X.1   +   1  1  1  1  1  1  1
X.2   +   1 -1  1  1 -1  1 -1
X.3   +   4  2  0  1  0 -1 -1
X.4   +   4 -2  0  1  0 -1  1
X.5   +   5  1  1 -1 -1  0  1
X.6   +   5 -1  1 -1  1  0 -1
X.7   +   6  0 -2  0  0  1  0



[]
>
> print CompositionFactors(S);
    G
    |  Cyclic(2)
    *
    |  Alternating(5)
```

```
    1
>
> ns:=NormalSubgroups(S); print [Order(ns[i]`subgroup) : i in [1..#ns]];
[ 1, 60, 120 ]
>
> ms:=MaximalSubgroups(S); print [Order(ms[i]`subgroup) : i in [1..#ms]];
[ 12, 20, 24, 60 ]
>
> quit;

Total time: 0.510 seconds, Total memory usage: 11.03MB


//////////////////////////
// Mathieu group M_{12}  //
//////////////////////////

**** Magma input ****

load m12;
print G;

print IsTransitive(G),IsPrimitive(G);
print Base(G);  BasicOrbitLengths(G);

print CompositionFactors(G);
print NormalSubgroups(G);

ms:=MaximalSubgroups(G);
print [Order(ms[i]`subgroup) : i in [1..#ms]];
print [Index(G,ms[i]`subgroup) : i in [1..#ms]];

print CharacterTable(G);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:08:06 on jones    [Seed = 1174705992]
Type ? for help.  Type <Ctrl>-D to quit.
>
> load m12;
Loading "/usr/local/magma/magma-2.17-7/libs/pergps/m12"
M12 - Mathieu group on 12 letters - degree 12
Order 95 040 = 2^6 * 3^3 * 5 * 11;  Base 1,2,3,4,5
Group: G
> print G;
Permutation group G acting on a set of cardinality 12
Order = 95040 = 2^6 * 3^3 * 5 * 11
    (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
    (1, 9)(2, 6)(4, 5)(7, 8)
    (1, 10)(2, 5)(3, 7)(4, 8)(6, 9)(11, 12)
>
> print IsTransitive(G),IsPrimitive(G);
true true
> print Base(G);  BasicOrbitLengths(G);
[ 12, 3, 1, 2, 4 ]
[ 12, 11, 10, 9, 8 ]
>
> print CompositionFactors(G);
    G
    |  M12
    1
> print NormalSubgroups(G);
Conjugacy classes of subgroups
------------------------------

[1]     Order 1           Length 1
        Permutation group acting on a set of cardinality 12
        Order = 1
```

```
            Id($)
[2]     Order 95040        Length 1
        Permutation group acting on a set of cardinality 12
        Order = 95040 = 2^6 * 3^3 * 5 * 11
            (1, 4)(2, 8)(3, 11)(5, 9)(6, 7)(10, 12)
            (1, 9, 7)(2, 4, 12)(3, 8, 11)
>
> ms:=MaximalSubgroups(G);
> print [Order(ms[i]`subgroup) : i in [1..#ms]];
[ 72, 660, 240, 192, 192, 432, 432, 1440, 1440, 7920, 7920 ]
> print [Index(G,ms[i]`subgroup) : i in [1..#ms]];
[ 1320, 144, 396, 495, 495, 220, 220, 66, 66, 12, 12 ]
>
> print CharacterTable(G);


Character Table of Group G
--------------------------


------------------------------------------------------------------------------
----
Class |     1   2   3    4    5    6    7    8    9    10    11    12   13
14
Size  |     1 396 495 1760 2640 2970 2970 9504 7920 15840 11880 11880 9504
8640
Order |     1   2   2    3    3    4    4    5    6     6     8     8   10
11
------------------------------------------------------------------------------
----
p  =  2     1   1   1    4    5    3    3    8    5     4     6     7    8
15
p  =  3     1   2   3    1    1    6    7    8    2     3    11    12   13
14
p  =  5     1   2   3    4    5    6    7    1    9    10    11    12    2
14
p  = 11     1   2   3    4    5    6    7    8    9    10    11    12   13
1
------------------------------------------------------------------------------
----
X.1   +     1   1   1    1    1    1    1    1    1     1     1     1    1
1
X.2   +    11  -1   3    2   -1   -1    3    1   -1     0    -1     1   -1
0
X.3   +    11  -1   3    2   -1    3   -1    1   -1     0     1    -1   -1
0
X.4   0    16   4   0   -2    1    0    0    1    1     0     0     0   -1
Z1
X.5   0    16   4   0   -2    1    0    0    1    1     0     0     0   -1
Z1#2
X.6   +    45   5  -3    0    3    1    1    0   -1     0    -1    -1    0
1
X.7   +    54   6   6    0    0    2    2   -1    0     0     0     0    1
-1
X.8   +    55  -5  -1    1    1    3   -1    0    1    -1    -1     1    0
0
X.9   +    55  -5   7    1    1   -1   -1    0    1     1    -1    -1    0
0
X.10  +    55  -5  -1    1    1   -1    3    0    1    -1     1    -1    0
0
X.11  +    66   6   2    3    0   -2   -2    1    0    -1     0     0    1
0
X.12  +    99  -1   3    0    3   -1   -1   -1   -1     0     1     1   -1
0
X.13  +   120   0  -8    3    0    0    0    0    0     1     0     0    0
-1
X.14  +   144   4   0    0   -3    0    0   -1    1     0     0     0   -1
1
```

```
X.15  +   176  -4   0   -4   -1    0    0    1   -1     0     0     0    1
0


--------------
Class |     15
Size  |   8640
Order |     11
--------------
p  =  2     14
p  =  3     15
p  =  5     15
p  = 11      1
--------------
X.1   +      1
X.2   +      0
X.3   +      0
X.4   0   Z1#2
X.5   0     Z1
X.6   +      1
X.7   +     -1
X.8   +      0
X.9   +      0
X.10  +      0
X.11  +      0
X.12  +      0
X.13  +     -1
X.14  +      1
X.15  +      0


Explanation of Character Value Symbols
--------------------------------------

# denotes algebraic conjugation, that is,
#k indicates replacing the root of unity w by w^k

Z1     = (CyclotomicField(11: Sparse := true)) ! [ RationalField() | -1,
-1, 0,
-1, -1, -1, 0, 0, 0, -1 ]


> quit;

Total time: 0.660 seconds, Total memory usage: 12.06MB


//////////////////////////
// Mathieu group M_{24}  //
//////////////////////////

**** Magma input ****

load m24;
print G;

print IsTransitive(G),IsPrimitive(G);
print Base(G);  BasicOrbitLengths(G);

print CompositionFactors(G);
print NormalSubgroups(G);

ms:=MaximalSubgroups(G);
print [Order(ms[i]`subgroup) : i in [1..#ms]];
print [Index(G,ms[i]`subgroup) : i in [1..#ms]];

**** Magma output ****
```

```
Magma V2.17-7     Fri Jul  1 2011 04:08:35 on jones    [Seed = 1511558065]
Type ? for help.  Type <Ctrl>-D to quit.
>
> load m24;
Loading "/usr/local/magma/magma-2.17-7/libs/pergps/m24"
M24 - Mathieu group on 24 letters - degree 24
Order 244 823 040 = 2^10 * 3^3 * 5 * 7 * 11 * 23;  Base 1,2,3,4,5,6,7
Group: G
> print G;
Permutation group G acting on a set of cardinality 24
Order = 244823040 = 2^10 * 3^3 * 5 * 7 * 11 * 23
    (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21,
        22, 24)
    (2, 16, 9, 6, 8)(3, 12, 13, 18, 4)(7, 17, 10, 11, 22)(14, 19, 21, 20,
    15)
    (1, 22)(2, 11)(3, 15)(4, 17)(5, 9)(6, 19)(7, 13)(8, 20)(10, 16)(12, 21)
    (14,
        18)(23, 24)
>
> print IsTransitive(G),IsPrimitive(G);
true true
> print Base(G);  BasicOrbitLengths(G);
[ 23, 1, 2, 3, 4, 5, 6 ]
[ 24, 23, 22, 21, 20, 16, 3 ]
>
> print CompositionFactors(G);
    G
    |  M24
    1
> print NormalSubgroups(G);
Conjugacy classes of subgroups
------------------------------

[1]     Order 1            Length 1
        Permutation group acting on a set of cardinality 24
        Order = 1
[2]     Order 244823040    Length 1
        Permutation group G acting on a set of cardinality 24
        Order = 244823040 = 2^10 * 3^3 * 5 * 7 * 11 * 23
            (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
            19,
                20, 21, 22, 24)
            (2, 16, 9, 6, 8)(3, 12, 13, 18, 4)(7, 17, 10, 11, 22)(14, 19,
            21,
                20, 15)
            (1, 22)(2, 11)(3, 15)(4, 17)(5, 9)(6, 19)(7, 13)(8, 20)(10, 16)
            (12,
                21)(14, 18)(23, 24)
>
> ms:=MaximalSubgroups(G);
> print [Order(ms[i]`subgroup) : i in [1..#ms]];
[ 168, 6072, 120960, 190080, 64512, 887040, 10200960, 138240, 322560 ]
> print [Index(G,ms[i]`subgroup) : i in [1..#ms]];
[ 1457280, 40320, 2024, 1288, 3795, 276, 24, 1771, 759 ]
>
> quit;

Total time: 0.530 seconds, Total memory usage: 14.19MB


////////////////////////////////////////////
// PSL(2,7) on 8 points, and action on cosets //
////////////////////////////////////////////

**** Magma input ****

P:=PSL(2,7);
```

```
print P;

print IsTransitive(P),IsPrimitive(P);

print Base(P);  BasicOrbitLengths(P);

cc:=ConjugacyClasses(P); print cc;
print CompositionFactors(P);
print NormalSubgroups(P);
print MaximalSubgroups(P);

ms:=MaximalSubgroups(P);
H:=ms[3]`subgroup;

f:=CosetAction(P,H);
Q:=Image(f); print Q;

print [Order(ms[i]`subgroup) : i in [1..#ms]];
print [Index(P,ms[i]`subgroup) : i in [1..#ms]];

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:09:05 on jones    [Seed = 1578930071]
Type ? for help.  Type <Ctrl>-D to quit.
>
> P:=PSL(2,7);
> print P;
Permutation group P acting on a set of cardinality 8
Order = 168 = 2^3 * 3 * 7
    (3, 6, 7)(4, 5, 8)
    (1, 8, 2)(4, 5, 6)
>
> print IsTransitive(P),IsPrimitive(P);
true true
>
> print Base(P);  BasicOrbitLengths(P);
[ 1, 3, 2 ]
[ 8, 7, 3 ]
>
> cc:=ConjugacyClasses(P); print cc;
Conjugacy Classes of group P
----------------------------
[1]     Order 1       Length 1
        Rep Id(P)

[2]     Order 2       Length 21
        Rep (1, 8)(2, 4)(3, 5)(6, 7)

[3]     Order 3       Length 56
        Rep (1, 8, 2)(4, 5, 6)

[4]     Order 4       Length 42
        Rep (1, 4, 8, 2)(3, 6, 5, 7)

[5]     Order 7       Length 24
        Rep (1, 5, 3, 7, 6, 8, 2)

[6]     Order 7       Length 24
        Rep (1, 7, 2, 3, 8, 5, 6)


> print CompositionFactors(P);
    G
    |  A(1, 7)               = L(2, 7)
    1
> print NormalSubgroups(P);
Conjugacy classes of subgroups
------------------------------
```

```
[1]     Order 1           Length 1
        Permutation group acting on a set of cardinality 8
        Order = 1
            Id($)
[2]     Order 168         Length 1
        Permutation group acting on a set of cardinality 8
        Order = 168 = 2^3 * 3 * 7
            (1, 8)(2, 4)(3, 5)(6, 7)
            (1, 6, 8, 3)(2, 5, 7, 4)
> print MaximalSubgroups(P);
Conjugacy classes of subgroups
------------------------------

[1]     Order 21          Length 8
        Permutation group acting on a set of cardinality 8
        Order = 21 = 3 * 7
            (1, 2, 3)(5, 7, 6)
            (1, 2, 5, 3, 6, 7, 4)
[2]     Order 24          Length 7
        Permutation group acting on a set of cardinality 8
        Order = 24 = 2^3 * 3
            (1, 7)(2, 5)(3, 6)(4, 8)
            (2, 3, 4)(6, 8, 7)
            (1, 3)(2, 4)(5, 6)(7, 8)
            (1, 2)(3, 4)(5, 7)(6, 8)
[3]     Order 24          Length 7
        Permutation group acting on a set of cardinality 8
        Order = 24 = 2^3 * 3
            (1, 6)(2, 4)(3, 7)(5, 8)
            (1, 3, 8)(4, 6, 7)
            (1, 2)(3, 8)(4, 7)(5, 6)
            (1, 3)(2, 8)(4, 5)(6, 7)
>
> ms:=MaximalSubgroups(P);
> H:=ms[3]`subgroup;
>
> f:=CosetAction(P,H);
> Q:=Image(f); print Q;
Permutation group Q acting on a set of cardinality 7
Order = 168 = 2^3 * 3 * 7
    (1, 2, 3)(5, 6, 7)
    (2, 4, 6)(3, 5, 7)
>
> print [Order(ms[i]`subgroup) : i in [1..#ms]];
[ 21, 24, 24 ]
> print [Index(P,ms[i]`subgroup) : i in [1..#ms]];
[ 8, 7, 7 ]
>
> quit;

Total time: 0.459 seconds, Total memory usage: 11.03MB


////////////////////////////////
// Transitive groups database  //
////////////////////////////////

**** Magma input ****

for n in [5..7] do
  tot:=NumberOfTransitiveGroups(n); print "";
  print "Degree",n,": Number of transitive groups =",tot;
  for i in [1..tot] do
    G:=TransitiveGroup(n,i);
    print i," |G| =",Order(G)," Primitive?",IsPrimitive(G);
    end for;
  end for;
```

```
**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:09:26 on jones    [Seed = 1376813964]
Type ? for help.  Type <Ctrl>-D to quit.
>
> for n in [5..7] do
for>   tot:=NumberOfTransitiveGroups(n); print "";
for>   print "Degree",n,": Number of transitive groups =",tot;
for>   for i in [1..tot] do
for|for>     G:=TransitiveGroup(n,i);
for|for>     print i," |G| =",Order(G)," Primitive?",IsPrimitive(G);
for|for>     end for;
for>   end for;

Degree 5 : Number of transitive groups = 5
1  |G| = 5  Primitive? true
2  |G| = 10  Primitive? true
3  |G| = 20  Primitive? true
4  |G| = 60  Primitive? true
5  |G| = 120  Primitive? true

Degree 6 : Number of transitive groups = 16
1  |G| = 6  Primitive? false
2  |G| = 6  Primitive? false
3  |G| = 12  Primitive? false
4  |G| = 12  Primitive? false
5  |G| = 18  Primitive? false
6  |G| = 24  Primitive? false
7  |G| = 24  Primitive? false
8  |G| = 24  Primitive? false
9  |G| = 36  Primitive? false
10  |G| = 36  Primitive? false
11  |G| = 48  Primitive? false
12  |G| = 60  Primitive? true
13  |G| = 72  Primitive? false
14  |G| = 120  Primitive? true
15  |G| = 360  Primitive? true
16  |G| = 720  Primitive? true

Degree 7 : Number of transitive groups = 7
1  |G| = 7  Primitive? true
2  |G| = 14  Primitive? true
3  |G| = 21  Primitive? true
4  |G| = 42  Primitive? true
5  |G| = 168  Primitive? true
6  |G| = 2520  Primitive? true
7  |G| = 5040  Primitive? true
>
> quit;

Total time: 0.399 seconds, Total memory usage: 9.69MB


////////////////////////////////
// Primitive groups database  //
////////////////////////////////

**** Magma input ****

for n in [41..44] do
  tot:=NumberOfPrimitiveGroups(n); print "";
  print "Degree",n,": Number of primitive groups =",tot;
  for i in [1..tot] do
    G:=PrimitiveGroup(n,i);
    cc:=ConjugacyClasses(G);
    // print i,Order(G),{g[1] : g in cc};
    print i," |G| =",Order(G)," Rank =",#Orbits(Stabilizer(G,1));
```

```
    end for;
  end for;

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:09:47 on jones    [Seed = 1444186101]
Type ? for help.  Type <Ctrl>-D to quit.
>
> for n in [41..44] do
for>   tot:=NumberOfPrimitiveGroups(n); print "";
for>   print "Degree",n,": Number of primitive groups =",tot;
for>   for i in [1..tot] do
for|for>     G:=PrimitiveGroup(n,i);
for|for>     cc:=ConjugacyClasses(G);
for|for>     // print i,Order(G),{g[1] : g in cc};
for|for>     print i," |G| =",Order(G)," Rank =",#Orbits(Stabilizer(G,1));
for|for>     end for;
for>   end for;

Degree 41 : Number of primitive groups = 10
1  |G| = 41  Rank = 41
2  |G| = 82  Rank = 21
3  |G| = 164  Rank = 11
4  |G| = 205  Rank = 9
5  |G| = 328  Rank = 6
6  |G| = 410  Rank = 5
7  |G| = 820  Rank = 3
8  |G| = 1640  Rank = 2
9  |G| = 167262633065819035540850310267203758325760000000000  Rank = 2
10  |G| = 33452526613163807108170062053440751665152000000000  Rank = 2

Degree 42 : Number of primitive groups = 4
1  |G| = 34440  Rank = 2
2  |G| = 68880  Rank = 2
3  |G| = 702503058876439949271571303122255784968192000000000  Rank = 2
4  |G| = 140500611775287989854314260624451156993638400000000  Rank = 2

Degree 43 : Number of primitive groups = 10
1  |G| = 43  Rank = 43
2  |G| = 86  Rank = 22
3  |G| = 129  Rank = 15
4  |G| = 258  Rank = 8
5  |G| = 301  Rank = 7
6  |G| = 602  Rank = 4
7  |G| = 903  Rank = 3
8  |G| = 1806  Rank = 2
9  |G| = 3020763153168691781867756603425699875363225600000000  Rank = 2
10  |G| = 6041526306337383563735513206851399750726451200000000  Rank = 2

Degree 44 : Number of primitive groups = 4
1  |G| = 39732  Rank = 2
2  |G| = 79464  Rank = 2
3  |G| = 1329135787394224384021812905507307945159819264000000000  Rank = 2
4  |G| = 2658271574788448768043625811014615890319638528000000000  Rank = 2
>
> quit;

Total time: 5.840 seconds, Total memory usage: 54.91MB


.....................................................................
Demonstration 5:  Use of MAGMA to study matrix groups
.....................................................................


//////////////
// GL_2(5)  //
//////////////
```

```
**** Magma input ****

G:=GL(2,5);
print G;

print Order(G);print FactoredOrder(G);

orbits:=Orbits(G);
print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
print "Orbit representatives",[Rep(B) : B in orbits];

print LineOrbits(G);
v:=Rep(orbits[2]); print v;print Stabilizer(G,v);

print CompositionFactors(G);

ns:=NormalSubgroups(G); print [Order(ns[i]`subgroup) : i in [1..#ns]];

IsIrreducible(G); MinimalField(G);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:10:21 on jones    [Seed = 711523247]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=GL(2,5);
> print G;
GL(2, GF(5))
>
> print Order(G);print FactoredOrder(G);
480
[ <2, 5>, <3, 1>, <5, 1> ]
>
> orbits:=Orbits(G);
> print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
Number of orbits = 2  Lengths = [ 1, 24 ]
> print "Orbit representatives",[Rep(B) : B in orbits];
Orbit representatives [
    (0 0),
    (1 0)
]
>
> print LineOrbits(G);
[
    {@
        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
        (1 0),

        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
        (1 4),

        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
        (0 1),

        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
        (1 2),

        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
        (1 1),

        Vector space of degree 2, dimension 1 over GF(5)
        Echelonized basis:
```

```
        (1 3)
    @}
]
> v:=Rep(orbits[2]); print v;print Stabilizer(G,v);
(1 0)
MatrixGroup(2, GF(5)) of order 2^2 * 5
Generators:
    [1 0]
    [2 1]

    [1 0]
    [0 3]
>
> print CompositionFactors(G);
    G
    |  Cyclic(2)
    *
    |  Alternating(5)
    *
    |  Cyclic(2)
    *
    |  Cyclic(2)
    1
>
> ns:=NormalSubgroups(G); print [Order(ns[i]`subgroup) : i in [1..#ns]];
[ 1, 2, 4, 120, 240, 480 ]
>
> IsIrreducible(G); MinimalField(G);
true
Finite field of size 5
>
> quit;

Total time: 0.450 seconds, Total memory usage: 11.03MB


//////////////
// SL_2(5)  //
//////////////

**** Magma input ****

G:=SL(2,5);
print Order(G);

orbits:=Orbits(G);
print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
v:=Rep(orbits[2]); print v;print Stabilizer(G,v);

print CharacterTable(G);

cc:=ConjugacyClasses(G);
for i in [1..#cc] do g:=cc[i][3];
  print i," Order",cc[i][1]," Class size",cc[i][2],
    " Trace",Trace(g)," MinPolynomial",MinimalPolynomial(g);
  end for;

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:10:53 on jones    [Seed = 778879051]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=SL(2,5);
> print Order(G);
120
>
> orbits:=Orbits(G);
> print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
```

```
Number of orbits = 2  Lengths = [ 1, 24 ]
> v:=Rep(orbits[2]); print v;print Stabilizer(G,v);
(1 0)
MatrixGroup(2, GF(5)) of order 5
Generators:
    [1 0]
    [2 1]
>
> print CharacterTable(G);


Character Table of Group G
--------------------------


-------------------------------------------
Class |   1  2  3  4    5    6  7    8    9
Size  |   1  1 20 30   12   12 20   12   12
Order |   1  2  3  4    5    5  6   10   10
-------------------------------------------
p  =  2   1  1  3  2    6    5  3    6    5
p  =  3   1  2  1  4    6    5  2    9    8
p  =  5   1  2  3  4    1    1  7    2    2
-------------------------------------------
X.1   +   1  1  1  1    1    1  1    1    1
X.2   -   2 -2 -1  0   Z1 Z1#2  1  -Z1-Z1#2
X.3   -   2 -2 -1  0 Z1#2   Z1  1-Z1#2  -Z1
X.4   +   3  3  0 -1-Z1#2  -Z1  0-Z1#2  -Z1
X.5   +   3  3  0 -1  -Z1-Z1#2  0  -Z1-Z1#2
X.6   +   4  4  1  0   -1   -1  1   -1   -1
X.7   -   4 -4  1  0   -1   -1 -1    1    1
X.8   +   5  5 -1  1    0    0 -1    0    0
X.9   -   6 -6  0  0    1    1  0   -1   -1


Explanation of Character Value Symbols
--------------------------------------

# denotes algebraic conjugation, that is,
#k indicates replacing the root of unity w by w^k

Z1     = (CyclotomicField(5: Sparse := true)) ! [ RationalField() | -1, 0,
-1,
-1 ]


[]
>
> cc:=ConjugacyClasses(G);
> for i in [1..#cc] do g:=cc[i][3];
for>   print i," Order",cc[i][1]," Class size",cc[i][2],
for|print>     " Trace",Trace(g)," MinPolynomial",MinimalPolynomial(g);
for>   end for;
1  Order 1  Class size 1  Trace 2  MinPolynomial $.1 + 4
2  Order 2  Class size 1  Trace 3  MinPolynomial $.1 + 1
3  Order 3  Class size 20  Trace 4  MinPolynomial $.1^2 + $.1 + 1
4  Order 4  Class size 30  Trace 0  MinPolynomial $.1^2 + 1
5  Order 5  Class size 12  Trace 2  MinPolynomial $.1^2 + 3*$.1 + 1
6  Order 5  Class size 12  Trace 2  MinPolynomial $.1^2 + 3*$.1 + 1
7  Order 6  Class size 20  Trace 1  MinPolynomial $.1^2 + 4*$.1 + 1
8  Order 10  Class size 12  Trace 3  MinPolynomial $.1^2 + 2*$.1 + 1
9  Order 10  Class size 12  Trace 3  MinPolynomial $.1^2 + 2*$.1 + 1
>
> quit;

Total time: 0.440 seconds, Total memory usage: 12.06MB
```

```
///////////////
// GL_2(27)  //
///////////////

**** Magma input ****

G:=GL(2,27);
print Order(G);

sgs:=Subgroups(G); print #sgs;
print #[i : i in [1..#sgs] | IsIrreducible(sgs[i]`subgroup)];
print {Order(sgs[i]`subgroup) : i in [1..#sgs]};
print {#Orbits(sgs[i]`subgroup) : i in [1..#sgs]};
print {MinimalField(sgs[i]`subgroup) : i in [1..#sgs]};

i:=Random([1..#sgs]);  H:=sgs[i]`subgroup;
print i,":",Order(H),IsIrreducible(H),#Orbits(H),#LineOrbits(H);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:11:25 on jones    [Seed = 576769127]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=GL(2,27);
> print Order(G);
511056
>
> sgs:=Subgroups(G); print #sgs;
192
> print #[i : i in [1..#sgs] | IsIrreducible(sgs[i]`subgroup)];
46
> print {Order(sgs[i]`subgroup) : i in [1..#sgs]};
{ 1, 2, 3, 4, 6, 7, 8, 9, 12, 13, 14, 16, 18, 24, 26, 27, 28, 36, 39, 48,
52,
54, 56, 78, 91, 104, 108, 112, 117, 156, 169, 182, 208, 234, 312, 338, 351,
364,
468, 624, 676, 702, 728, 1352, 1404, 1456, 4563, 9126, 18252, 19656, 39312,
255528, 511056 }
> print {#Orbits(sgs[i]`subgroup) : i in [1..#sgs]};
{ 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 14, 15, 16, 21, 27, 28, 29, 30, 40, 41,
42,
53, 54, 57, 66, 79, 81, 92, 105, 131, 144, 183, 196, 261, 365, 378, 729 }
> print {MinimalField(sgs[i]`subgroup) : i in [1..#sgs]};
{
    Finite field of size 3,
    Finite field of size 3^3
}
>
> i:=Random([1..#sgs]);  H:=sgs[i]`subgroup;
> print i,":",Order(H),IsIrreducible(H),#Orbits(H),#LineOrbits(H);
18 : 6 false 144 6
>
> quit;

Total time: 1.770 seconds, Total memory usage: 11.03MB


//////////////
// SO_5(3)  //
//////////////

**** Magma input ****

G:=SpecialOrthogonalGroup(5,3);
print Order(G);

orbits:=Orbits(G);
print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
```

```
print "Orbit representatives",[Rep(B) : B in orbits];

print CompositionFactors(G);

ns:=NormalSubgroups(G); print [Order(ns[i]`subgroup) : i in [1..#ns]];
IsIrreducible(G);

sgs:=Subgroups(G); print #sgs;
print #[i : i in [1..#sgs] | IsIrreducible(sgs[i]`subgroup)];
print {Order(sgs[i]`subgroup) : i in [1..#sgs]};
print {#Orbits(sgs[i]`subgroup) : i in [1..#sgs]};

i:=Random([1..#sgs]);  H:=sgs[i]`subgroup;
print i,":",Order(H),IsIrreducible(H),#Orbits(H),#LineOrbits(H);
print CharacterTable(H);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:11:55 on jones    [Seed = 1048367133]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=SpecialOrthogonalGroup(5,3);
> print Order(G);
51840
>
> orbits:=Orbits(G);
> print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
Number of orbits = 4  Lengths = [ 1, 80, 90, 72 ]
> print "Orbit representatives",[Rep(B) : B in orbits];
Orbit representatives [
    (0 0 0 0 0),
    (1 0 0 0 0),
    (0 0 1 0 0),
    (0 2 0 1 0)
]
>
> print CompositionFactors(G);
    G
    |  Cyclic(2)
    *
    |  C(2, 3)               = S(4, 3)
    1
>
> ns:=NormalSubgroups(G); print [Order(ns[i]`subgroup) : i in [1..#ns]];
[ 1, 25920, 51840 ]
> IsIrreducible(G);
true
>
> sgs:=Subgroups(G); print #sgs;
350
> print #[i : i in [1..#sgs] | IsIrreducible(sgs[i]`subgroup)];
7
> print {Order(sgs[i]`subgroup) : i in [1..#sgs]};
{ 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 18, 20, 24, 27, 32, 36, 40, 48, 54,
60,
64, 72, 80, 81, 96, 108, 120, 128, 144, 160, 162, 192, 216, 240, 288, 320,
324,
360, 384, 432, 576, 648, 720, 960, 1152, 1296, 1440, 1920, 25920, 51840 }
> print {#Orbits(sgs[i]`subgroup) : i in [1..#sgs]};
{ 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
27, 29, 30, 31, 33, 35, 36, 39, 41, 42, 45, 48, 51, 54, 55, 63, 69, 75, 81,
99,
123, 135, 243 }
>
> i:=Random([1..#sgs]);  H:=sgs[i]`subgroup;
> print i,":",Order(H),IsIrreducible(H),#Orbits(H),#LineOrbits(H);
80 : 12 false 33 31
```

```
> print CharacterTable(H);


Character Table of Group H
--------------------------


-------------------------
Class |   1  2  3  4  5  6
Size  |   1  1  3  3  2  2
Order |   1  2  2  2  3  6
-------------------------
p  =  2   1  1  1  1  5  5
p  =  3   1  2  3  4  1  2
-------------------------
X.1   +   1  1  1  1  1  1
X.2   +   1  1 -1 -1  1  1
X.3   +   1 -1  1 -1  1 -1
X.4   +   1 -1 -1  1  1 -1
X.5   +   2  2  0  0 -1 -1
X.6   +   2 -2  0  0 -1  1



[]
>
> quit;

Total time: 0.860 seconds, Total memory usage: 15.16MB


//////////////////////////
// Suzuki group Sz(32) //
//////////////////////////

**** Magma input ****

G:=SuzukiGroup(2^5);
print Order(G);

orbits:=Orbits(G);
print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
print "Orbit representatives",[Rep(B) : B in orbits];

print CompositionFactors(G);

cc:=ConjugacyClasses(G);
for i in [1..#cc] do g:=cc[i][3];
  print i," Order",cc[i][1]," Class size",cc[i][2],
    " Trace",Trace(g)," MinPolynomial",MinimalPolynomial(g);
  end for;

print CharacterTable(G);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:12:27 on jones    [Seed = 846257201]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=SuzukiGroup(2^5);
> print Order(G);
32537600
>
> orbits:=Orbits(G);
> print "Number of orbits =",#orbits," Lengths =",[#B : B in orbits];
Number of orbits = 3  Lengths = [ 31775, 1016800, 1 ]
> print "Orbit representatives",[Rep(B) : B in orbits];
Orbit representatives [
```

```
    (     1      1      1      1),
    (   $.1      1      1      1),
    (     0      0      0      0)
]
>
> print CompositionFactors(G);
    G
    |  2B(2, 32)             = Sz(32)
    1
>
> cc:=ConjugacyClasses(G);
> for i in [1..#cc] do g:=cc[i][3];
for>   print i," Order",cc[i][1]," Class size",cc[i][2],
for|print>     " Trace",Trace(g)," MinPolynomial",MinimalPolynomial(g);
for>   end for;
1  Order 1  Class size 1  Trace 0  MinPolynomial $.1 + 1
2  Order 2  Class size 31775  Trace 0  MinPolynomial $.1^2 + 1
3  Order 4  Class size 508400  Trace 0  MinPolynomial $.1^4 + 1
4  Order 4  Class size 508400  Trace 0  MinPolynomial $.1^4 + 1
5  Order 5  Class size 1301504  Trace 1  MinPolynomial $.1^4 + $.1^3 +
$.1^2 +
    $.1 + 1
6  Order 25  Class size 1301504  Trace $.1^4  MinPolynomial $.1^4 +
$.1^4*$.1^3
    + $.1*$.1^2 + $.1^4*$.1 + 1
7  Order 25  Class size 1301504  Trace $.1^16  MinPolynomial $.1^4 +
    $.1^16*$.1^3 + $.1^4*$.1^2 + $.1^16*$.1 + 1
8  Order 25  Class size 1301504  Trace $.1  MinPolynomial $.1^4 + $.1*$.1^3
+
    $.1^8*$.1^2 + $.1*$.1 + 1
9  Order 25  Class size 1301504  Trace $.1^2  MinPolynomial $.1^4 +
$.1^2*$.1^3
    + $.1^16*$.1^2 + $.1^2*$.1 + 1
10  Order 25  Class size 1301504  Trace $.1^8  MinPolynomial $.1^4 +
$.1^8*$.1^3
    + $.1^2*$.1^2 + $.1^8*$.1 + 1
11  Order 31  Class size 1049600  Trace $.1^11  MinPolynomial $.1^4 +
    $.1^11*$.1^3 + $.1^26*$.1^2 + $.1^11*$.1 + 1
12  Order 31  Class size 1049600  Trace $.1^26  MinPolynomial $.1^4 +
    $.1^26*$.1^3 + $.1^22*$.1^2 + $.1^26*$.1 + 1
13  Order 31  Class size 1049600  Trace $.1^23  MinPolynomial $.1^4 +
    $.1^23*$.1^3 + $.1^29*$.1^2 + $.1^23*$.1 + 1
14  Order 31  Class size 1049600  Trace $.1^21  MinPolynomial $.1^4 +
    $.1^21*$.1^3 + $.1^13*$.1^2 + $.1^21*$.1 + 1
15  Order 31  Class size 1049600  Trace $.1^10  MinPolynomial $.1^4 +
    $.1^10*$.1^3 + $.1^18*$.1^2 + $.1^10*$.1 + 1
16  Order 31  Class size 1049600  Trace $.1^15  MinPolynomial $.1^4 +
    $.1^15*$.1^3 + $.1^27*$.1^2 + $.1^15*$.1 + 1
17  Order 31  Class size 1049600  Trace $.1^29  MinPolynomial $.1^4 +
    $.1^29*$.1^3 + $.1^15*$.1^2 + $.1^29*$.1 + 1
18  Order 31  Class size 1049600  Trace $.1^13  MinPolynomial $.1^4 +
    $.1^13*$.1^3 + $.1^11*$.1^2 + $.1^13*$.1 + 1
19  Order 31  Class size 1049600  Trace $.1^18  MinPolynomial $.1^4 +
    $.1^18*$.1^3 + $.1^20*$.1^2 + $.1^18*$.1 + 1
20  Order 31  Class size 1049600  Trace $.1^20  MinPolynomial $.1^4 +
    $.1^20*$.1^3 + $.1^5*$.1^2 + $.1^20*$.1 + 1
21  Order 31  Class size 1049600  Trace $.1^9  MinPolynomial $.1^4 +
$.1^9*$.1^3
    + $.1^10*$.1^2 + $.1^9*$.1 + 1
22  Order 31  Class size 1049600  Trace $.1^30  MinPolynomial $.1^4 +
    $.1^30*$.1^3 + $.1^23*$.1^2 + $.1^30*$.1 + 1
23  Order 31  Class size 1049600  Trace $.1^5  MinPolynomial $.1^4 +
$.1^5*$.1^3
    + $.1^9*$.1^2 + $.1^5*$.1 + 1
24  Order 31  Class size 1049600  Trace $.1^27  MinPolynomial $.1^4 +
    $.1^27*$.1^3 + $.1^30*$.1^2 + $.1^27*$.1 + 1
25  Order 31  Class size 1049600  Trace $.1^22  MinPolynomial $.1^4 +
    $.1^22*$.1^3 + $.1^21*$.1^2 + $.1^22*$.1 + 1
```

```
26  Order 41  Class size 793600  Trace $.1^6  MinPolynomial $.1^4 +
$.1^6*$.1^3
    + $.1^17*$.1^2 + $.1^6*$.1 + 1
27  Order 41  Class size 793600  Trace $.1^12  MinPolynomial $.1^4 +
    $.1^12*$.1^3 + $.1^3*$.1^2 + $.1^12*$.1 + 1
28  Order 41  Class size 793600  Trace $.1^19  MinPolynomial $.1^4 +
    $.1^19*$.1^3 + $.1^28*$.1^2 + $.1^19*$.1 + 1
29  Order 41  Class size 793600  Trace $.1^24  MinPolynomial $.1^4 +
    $.1^24*$.1^3 + $.1^6*$.1^2 + $.1^24*$.1 + 1
30  Order 41  Class size 793600  Trace $.1^7  MinPolynomial $.1^4 +
$.1^7*$.1^3
    + $.1^25*$.1^2 + $.1^7*$.1 + 1
31  Order 41  Class size 793600  Trace $.1^25  MinPolynomial $.1^4 +
    $.1^25*$.1^3 + $.1^14*$.1^2 + $.1^25*$.1 + 1
32  Order 41  Class size 793600  Trace $.1^17  MinPolynomial $.1^4 +
    $.1^17*$.1^3 + $.1^12*$.1^2 + $.1^17*$.1 + 1
33  Order 41  Class size 793600  Trace $.1^28  MinPolynomial $.1^4 +
    $.1^28*$.1^3 + $.1^7*$.1^2 + $.1^28*$.1 + 1
34  Order 41  Class size 793600  Trace $.1^14  MinPolynomial $.1^4 +
    $.1^14*$.1^3 + $.1^19*$.1^2 + $.1^14*$.1 + 1
35  Order 41  Class size 793600  Trace $.1^3  MinPolynomial $.1^4 +
$.1^3*$.1^3
    + $.1^24*$.1^2 + $.1^3*$.1 + 1
>
> print CharacterTable(G);


Character Table of Group G
--------------------------


-------------------------------------------------------------------------------
Class |      1     2      3      4       5       6       7       8       9
Size  |      1 31775 508400 508400 1301504 1301504 1301504 1301504 1301504
Order |      1     2      4      4       5      25      25      25      25
-------------------------------------------------------------------------------
p  =  2      1     1      2      2       5      10       8       9       6
p  =  5      1     2      3      4       1       5       5       5       5
p  = 31      1     2      4      3       5       8       6      10       7
p  = 41      1     2      3      4       5       9      10       7       8
-------------------------------------------------------------------------------
X.1   +      1     1      1      1       1       1       1       1       1
X.2   0    124    -4   -4*I    4*I      -1      -1      -1      -1      -1
X.3   0    124    -4    4*I   -4*I      -1      -1      -1      -1      -1
X.4   +    775     7     -1     -1       0       0       0       0       0
X.5   +    775     7     -1     -1       0       0       0       0       0
X.6   +    775     7     -1     -1       0       0       0       0       0
X.7   +    775     7     -1     -1       0       0       0       0       0
X.8   +    775     7     -1     -1       0       0       0       0       0
X.9   +    775     7     -1     -1       0       0       0       0       0
X.10  +    775     7     -1     -1       0       0       0       0       0
X.11  +    775     7     -1     -1       0       0       0       0       0
X.12  +    775     7     -1     -1       0       0       0       0       0
X.13  +    775     7     -1     -1       0       0       0       0       0
X.14  +   1024     0      0      0      -1      -1      -1      -1      -1
X.15  +   1025     1      1      1       0       0       0       0       0
X.16  +   1025     1      1      1       0       0       0       0       0
X.17  +   1025     1      1      1       0       0       0       0       0
X.18  +   1025     1      1      1       0       0       0       0       0
X.19  +   1025     1      1      1       0       0       0       0       0
X.20  +   1025     1      1      1       0       0       0       0       0
X.21  +   1025     1      1      1       0       0       0       0       0
X.22  +   1025     1      1      1       0       0       0       0       0
X.23  +   1025     1      1      1       0       0       0       0       0
X.24  +   1025     1      1      1       0       0       0       0       0
X.25  +   1025     1      1      1       0       0       0       0       0
X.26  +   1025     1      1      1       0       0       0       0       0
X.27  +   1025     1      1      1       0       0       0       0       0
```

X.28	+	1025	1	1	1	0	0	0	0	0
X.29	+	1025	1	1	1	0	0	0	0	0
X.30	+	1271	-9	-1	-1	-4	1	1	1	1
X.31	+	1271	-9	-1	-1	1	Z1	Z1#3	Z1#6	Z1#9
X.32	+	1271	-9	-1	-1	1	Z1#3	Z1#9	Z1	Z1#2
X.33	+	1271	-9	-1	-1	1	Z1#6	Z1	Z1#2	Z1#3
X.34	+	1271	-9	-1	-1	1	Z1#9	Z1#2	Z1#3	Z1#6
X.35	+	1271	-9	-1	-1	1	Z1#2	Z1#6	Z1#9	Z1

Class		10	11	12	13	14	15	16	17
Size		1301504	1049600	1049600	1049600	1049600	1049600	1049600	1049600
Order		25	31	31	31	31	31	31	31
p = 2		7	25	14	16	11	20	22	24
p = 5		5	19	20	25	21	16	18	14
p = 31		9	1	1	1	1	1	1	1
p = 41		6	23	21	18	19	22	12	11
X.1	+	1	1	1	1	1	1	1	1
X.2	0	-1	0	0	0	0	0	0	0
X.3	0	-1	0	0	0	0	0	0	0
X.4	+	0	0	0	0	0	0	0	0
X.5	+	0	0	0	0	0	0	0	0
X.6	+	0	0	0	0	0	0	0	0
X.7	+	0	0	0	0	0	0	0	0
X.8	+	0	0	0	0	0	0	0	0
X.9	+	0	0	0	0	0	0	0	0
X.10	+	0	0	0	0	0	0	0	0
X.11	+	0	0	0	0	0	0	0	0
X.12	+	0	0	0	0	0	0	0	0
X.13	+	0	0	0	0	0	0	0	0
X.14	+	-1	1	1	1	1	1	1	1
X.15	+	0	Z2	Z2#8	Z2#12	Z2#15	Z2#11	Z2#7	Z2#3
X.16	+	0	Z2#15	Z2#4	Z2#6	Z2#8	Z2#10	Z2#12	Z2#14
X.17	+	0	Z2#3	Z2#7	Z2#5	Z2#14	Z2#2	Z2#10	Z2#9
X.18	+	0	Z2#11	Z2#5	Z2#8	Z2#10	Z2#3	Z2#15	Z2#2
X.19	+	0	Z2#10	Z2#13	Z2#4	Z2#5	Z2#14	Z2#8	Z2
X.20	+	0	Z2#4	Z2	Z2#14	Z2#2	Z2#13	Z2#3	Z2#12
X.21	+	0	Z2#12	Z2#3	Z2#11	Z2#6	Z2#8	Z2#9	Z2#5
X.22	+	0	Z2#9	Z2#10	Z2#15	Z2#11	Z2#6	Z2	Z2#4
X.23	+	0	Z2#5	Z2#9	Z2#2	Z2#13	Z2#7	Z2#4	Z2#15
X.24	+	0	Z2#7	Z2#6	Z2#9	Z2#12	Z2#15	Z2#13	Z2#10
X.25	+	0	Z2#8	Z2#2	Z2#3	Z2#4	Z2#5	Z2#6	Z2#7
X.26	+	0	Z2#2	Z2#15	Z2#7	Z2	Z2#9	Z2#14	Z2#6
X.27	+	0	Z2#14	Z2#12	Z2#13	Z2#7	Z2	Z2#5	Z2#11
X.28	+	0	Z2#6	Z2#14	Z2#10	Z2#3	Z2#4	Z2#11	Z2#13
X.29	+	0	Z2#13	Z2#11	Z2	Z2#9	Z2#12	Z2#2	Z2#8
X.30	+	1	0	0	0	0	0	0	0
X.31	+	Z1#2	0	0	0	0	0	0	0
X.32	+	Z1#6	0	0	0	0	0	0	0
X.33	+	Z1#9	0	0	0	0	0	0	0
X.34	+	Z1	0	0	0	0	0	0	0
X.35	+	Z1#3	0	0	0	0	0	0	0

Class		18	19	20	21	22	23	24	25
Size		1049600	1049600	1049600	1049600	1049600	1049600	1049600	1049600
Order		31	31	31	31	31	31	31	31
p = 2		12	23	21	19	17	15	13	18
p = 5		15	24	22	17	12	13	11	23
p = 31		1	1	1	1	1	1	1	1
p = 41		20	13	17	24	14	16	25	15
X.1	+	1	1	1	1	1	1	1	1

```
X.2   0         0       0       0       0       0       0       0       0
X.3   0         0       0       0       0       0       0       0       0
X.4   +         0       0       0       0       0       0       0       0
X.5   +         0       0       0       0       0       0       0       0
X.6   +         0       0       0       0       0       0       0       0
X.7   +         0       0       0       0       0       0       0       0
X.8   +         0       0       0       0       0       0       0       0
X.9   +         0       0       0       0       0       0       0       0
X.10  +         0       0       0       0       0       0       0       0
X.11  +         0       0       0       0       0       0       0       0
X.12  +         0       0       0       0       0       0       0       0
X.13  +         0       0       0       0       0       0       0       0
X.14  +         1       1       1       1       1       1       1       1
X.15  +      Z2#4    Z2#5    Z2#9   Z2#13   Z2#14   Z2#10    Z2#6    Z2#2
X.16  +      Z2#2   Z2#13   Z2#11    Z2#9    Z2#7    Z2#5    Z2#3      Z2
X.17  +     Z2#12   Z2#15    Z2#4    Z2#8   Z2#11      Z2   Z2#13    Z2#6
X.18  +     Z2#13    Z2#7    Z2#6   Z2#12      Z2   Z2#14    Z2#4    Z2#9
X.19  +      Z2#9   Z2#12    Z2#3    Z2#6   Z2#15    Z2#7    Z2#2   Z2#11
X.20  +     Z2#15   Z2#11    Z2#5   Z2#10    Z2#6    Z2#9    Z2#7    Z2#8
X.21  +     Z2#14    Z2#2   Z2#15      Z2   Z2#13    Z2#4   Z2#10    Z2#7
X.22  +      Z2#5   Z2#14   Z2#12    Z2#7    Z2#2    Z2#3    Z2#8   Z2#13
X.23  +     Z2#11    Z2#6   Z2#14    Z2#3    Z2#8   Z2#12      Z2   Z2#10
X.24  +      Z2#3    Z2#4      Z2    Z2#2    Z2#5    Z2#8   Z2#11   Z2#14
X.25  +        Z2    Z2#9   Z2#10   Z2#11   Z2#12   Z2#13   Z2#14   Z2#15
X.26  +      Z2#8   Z2#10   Z2#13    Z2#5    Z2#3   Z2#11   Z2#12    Z2#4
X.27  +      Z2#6    Z2#8    Z2#2    Z2#4   Z2#10   Z2#15    Z2#9    Z2#3
X.28  +      Z2#7      Z2    Z2#8   Z2#15    Z2#9    Z2#2    Z2#5   Z2#12
X.29  +     Z2#10    Z2#3    Z2#7   Z2#14    Z2#4    Z2#6   Z2#15    Z2#5
X.30  +         0       0       0       0       0       0       0       0
X.31  +         0       0       0       0       0       0       0       0
X.32  +         0       0       0       0       0       0       0       0
X.33  +         0       0       0       0       0       0       0       0
X.34  +         0       0       0       0       0       0       0       0
X.35  +         0       0       0       0       0       0       0       0


-------------------------------------------------------------------------
----
Class |       26     27     28     29     30     31     32     33     34
35
Size  |   793600 793600 793600 793600 793600 793600 793600 793600 793600
793600
Order |       41     41     41     41     41     41     41     41     41
41
-------------------------------------------------------------------------
----
p  =  2       27     29     30     32     34     28     35     31     33
26
p  =  5       29     32     34     35     33     30     26     28     31
27
p  = 31       32     35     33     26     31     34     27     30     28
29
p  = 41        1      1      1      1      1      1      1      1      1
1
-------------------------------------------------------------------------
----
X.1   +        1      1      1      1      1      1      1      1      1
1
X.2   0        1      1      1      1      1      1      1      1      1
1
X.3   0        1      1      1      1      1      1      1      1      1
1
X.4   +       Z3   Z3#2   Z3#3   Z3#4   Z3#6   Z3#7   Z3#8  Z3#11  Z3#12
Z3#16
X.5   +     Z3#2   Z3#4   Z3#6   Z3#8  Z3#12   Z3#3  Z3#16   Z3#7  Z3#11
Z3
X.6   +    Z3#16     Z3   Z3#7   Z3#2   Z3#3  Z3#11   Z3#4  Z3#12   Z3#6
Z3#8
```

```
X.7   +     Z3#8  Z3#16  Z3#11     Z3   Z3#7  Z3#12   Z3#2   Z3#6   Z3#3
Z3#4
X.8   +     Z3#4   Z3#8  Z3#12  Z3#16  Z3#11   Z3#6     Z3   Z3#3   Z3#7
Z3#2
X.9   +     Z3#6  Z3#12   Z3#2  Z3#11   Z3#4     Z3   Z3#7  Z3#16   Z3#8
Z3#3
X.10  +    Z3#11   Z3#7   Z3#8   Z3#3  Z3#16   Z3#4   Z3#6   Z3#2     Z3
Z3#12
X.11  +     Z3#7   Z3#3  Z3#16   Z3#6     Z3   Z3#8  Z3#12   Z3#4   Z3#2
Z3#11
X.12  +    Z3#12  Z3#11   Z3#4   Z3#7   Z3#8   Z3#2   Z3#3     Z3  Z3#16
Z3#6
X.13  +     Z3#3   Z3#6     Z3  Z3#12   Z3#2  Z3#16  Z3#11   Z3#8   Z3#4
Z3#7
X.14  +       -1     -1     -1     -1     -1     -1     -1     -1     -1
-1
X.15  +        0      0      0      0      0      0      0      0      0
0
X.16  +        0      0      0      0      0      0      0      0      0
0
X.17  +        0      0      0      0      0      0      0      0      0
0
X.18  +        0      0      0      0      0      0      0      0      0
0
X.19  +        0      0      0      0      0      0      0      0      0
0
X.20  +        0      0      0      0      0      0      0      0      0
0
X.21  +        0      0      0      0      0      0      0      0      0
0
X.22  +        0      0      0      0      0      0      0      0      0
0
X.23  +        0      0      0      0      0      0      0      0      0
0
X.24  +        0      0      0      0      0      0      0      0      0
0
X.25  +        0      0      0      0      0      0      0      0      0
0
X.26  +        0      0      0      0      0      0      0      0      0
0
X.27  +        0      0      0      0      0      0      0      0      0
0
X.28  +        0      0      0      0      0      0      0      0      0
0
X.29  +        0      0      0      0      0      0      0      0      0
0
X.30  +        0      0      0      0      0      0      0      0      0
0
X.31  +        0      0      0      0      0      0      0      0      0
0
X.32  +        0      0      0      0      0      0      0      0      0
0
X.33  +        0      0      0      0      0      0      0      0      0
0
X.34  +        0      0      0      0      0      0      0      0      0
0
X.35  +        0      0      0      0      0      0      0      0      0
0


Explanation of Character Value Symbols
--------------------------------------

# denotes algebraic conjugation, that is,
#k indicates replacing the root of unity w by w^k

I = RootOfUnity(4)
```

```
Z1     = (CyclotomicField(25: Sparse := true)) ! [ RationalField() | 0, 0,
0, 0,
0, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1 ]

Z2     = (CyclotomicField(31: Sparse := true)) ! [ RationalField() | 0, 0,
0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0 ]

Z3     = (CyclotomicField(41: Sparse := true)) ! [ RationalField() | 0, 0,
-1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 0, 0, 0,
0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1 ]


> quit;

Total time: 6.790 seconds, Total memory usage: 123.41MB


////////////////////////////
// Recognising a subgroup //
////////////////////////////

**** Magma input ****

G:=SL(3,29);

x:=G![ 8, 26, 13, 18, 22, 26, 6, 27, 27 ];print x;
y:=G![ 26, 15, 5, 15, 6, 10, 5, 10, 26 ];print y;

H:=sub<G|x,y>; print Order(H);

print CompositionFactors(H);

z:=Random(G);
K:=sub<G|x,z>; print Order(K);
print CompositionFactors(K);

**** Magma output ****

Magma V2.17-7     Fri Jul  1 2011 04:13:05 on jones    [Seed = 913627343]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G:=SL(3,29);
>
> x:=G![ 8, 26, 13, 18, 22, 26, 6, 27, 27 ];print x;
[ 8 26 13]
[18 22 26]
[ 6 27 27]
> y:=G![ 26, 15, 5, 15, 6, 10, 5, 10, 26 ];print y;
[26 15  5]
[15  6 10]
[ 5 10 26]
>
> H:=sub<G|x,y>; print Order(H);
168
>
> print CompositionFactors(H);
    G
    |  A(1, 7)               = L(2, 7)
    1
>
> z:=Random(G);
> K:=sub<G|x,z>; print Order(K);
499631102880
> print CompositionFactors(K);
```

```
    G
    |  A(2, 29)              = L(3, 29)
    1
>
> quit;

Total time: 1.090 seconds, Total memory usage: 10.56MB


////////////////////////////////////////////////////////////////////////
// Lecture 3 ///////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////


.........................................................................
..
Demonstration 6:  Coset enumeration
.........................................................................
..

////////////////////////
// Dihedral group D_5 //
////////////////////////

**** Magma input ****

G<x,y>:=Group<x,y| x^2, y^5, (x*y)^2 >;

print Order(G);

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 19:54:52 on mcon007-01 [Seed =
1595698490]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<x,y>:=Group<x,y| x^2, y^5, (x*y)^2 >;
>
> print Order(G);
10
>
> quit;

Total time: 0.410 seconds, Total memory usage: 22.47MB


//////////////////////////
// Free group of rank 3 //
//////////////////////////

**** Magma input ****

F:=FreeGroup(3);
print Order(F);

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 19:55:14 on mcon007-01 [Seed =
1393582360]
Type ? for help.  Type <Ctrl>-D to quit.
>
> F:=FreeGroup(3);
> print Order(F);
Infinity
>
> quit;

Total time: 0.250 seconds, Total memory usage: 7.29MB
```

```
//////////////////////////////
// Spherical triangle groups //
//////////////////////////////

**** Magma input ****

for p in [2..6] do for q in [p..6] do for r in [q..6] do
  if (1/p)+(1/q)+(1/r) gt 1 then
    G<x,y,z>:=Group<x,y,z| x*y*z, x^p, y^q, z^r >;
    print p,q,r,":",Order(G);
    end if;
  end for;end for;end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 19:55:45 on mcon007-01 [Seed =
1444242686]
Type ? for help.  Type <Ctrl>-D to quit.
>
> for p in [2..6] do for q in [p..6] do for r in [q..6] do
for|for|for>   if (1/p)+(1/q)+(1/r) gt 1 then
for|for|for|if>     G<x,y,z>:=Group<x,y,z| x*y*z, x^p, y^q, z^r >;
for|for|for|if>     print p,q,r,":",Order(G);
for|for|for|if>     end if;
for|for|for>   end for;end for;end for;
2 2 2 : 4
2 2 3 : 6
2 2 4 : 8
2 2 5 : 10
2 2 6 : 12
2 3 3 : 12
2 3 4 : 24
2 3 5 : 60
>
> quit;

Total time: 0.329 seconds, Total memory usage: 22.57MB


////////////////////////////////////
// Modular group C2 * C3 = PSL(2,Z) //
////////////////////////////////////

**** Magma input ****

G<x,y>:=Group< x,y | x^2, y^3 >;
print AbelianQuotient(G);
print AbelianQuotientInvariants(G);

H:=sub<G| x^-1*y^-1*x*y, x*y*x^-1*y^-1 >;
print Index(G,H);

print CosetTable(G,H);
f:=CosetAction(G,H); P:=Image(f);  print P;
S:=SchreierGraph(G,H); print S; Edges(S);

print Order(G);

K:=Rewrite(G,H); print K;
print AbelianQuotientInvariants(K);

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 19:56:15 on mcon007-01 [Seed =
2867435548]
Type ? for help.  Type <Ctrl>-D to quit.
```

```
>
> G<x,y>:=Group< x,y | x^2, y^3 >;
> print AbelianQuotient(G);
Abelian Group isomorphic to Z/6
Defined on 1 generator
Relations:
    6*$.1 = 0
> print AbelianQuotientInvariants(G);
[ 6 ]
>
> H:=sub<G| x^-1*y^-1*x*y, x*y*x^-1*y^-1 >;
> print Index(G,H);
6
>
> print CosetTable(G,H);
Mapping from: Cartesian Product<{ 1 .. 6 }, GrpFP: G> to { 1 .. 6 }
    $1  $2 -$2
1.   2   3   4
2.   1   5   6
3.   5   4   1
4.   6   1   3
5.   3   6   2
6.   4   2   5

> f:=CosetAction(G,H); P:=Image(f);  print P;
Permutation group P acting on a set of cardinality 6
    (1, 2)(3, 5)(4, 6)
    (1, 3, 4)(2, 5, 6)
> S:=SchreierGraph(G,H); print S; Edges(S);
Digraph
Vertex  Neighbours

1       2 3 ;
2       1 5 ;
3       4 5 ;
4       1 6 ;
5       3 6 ;
6       2 4 ;

{@ [1, 2], [1, 3], [2, 1], [2, 5], [3, 4], [3, 5], [4, 1], [4, 6], [5, 3],
[5,
6], [6, 2], [6, 4] @}
>
> print Order(G);
0
>
> K:=Rewrite(G,H); print K;
Finitely presented group K on 2 generators
Generators as words in group G
    K.1 = y * x * y^-1 * x
    K.2 = y^-1 * x * y * x
> print AbelianQuotientInvariants(K);
[ 0, 0 ]
>
> quit;

Total time: 0.490 seconds, Total memory usage: 22.61MB


//////////////////////
// Mystery example //
//////////////////////

**** Magma input ****

G<a,b,c>:=Group<a,b,c| a^11, b^5, c^4, (a*c)^7, (a*c^-1)^7,
 b^-1*a*b*a^-3, c^-1*b*c*b^-2, a*c^2*a*c^2*a*c^-2,
 (a*c*a^-1*c*a*b^-1*c^-1)^2, (a*c^-1*a^-1*c^-1*a*c*b)^2,
```

```
 a*c*a^2*c^-2*a^-1*c*a^-2*c*a^2*b*c^-1 >;

print Order(G);

n,ct:=ToddCoxeter(G,sub<G|>: CosetLimit:=500000);
print n;

Q:=CosetImage(G,sub<G|a>);
print Degree(Q),Order(Q); print CompositionFactors(Q);

n,ct:=ToddCoxeter(G,sub<G|>: CosetLimit:=2000000);
print n;


**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 19:57:28 on mcon007-01 [Seed =
2816775526]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<a,b,c>:=Group<a,b,c| a^11, b^5, c^4, (a*c)^7, (a*c^-1)^7,
>  b^-1*a*b*a^-3, c^-1*b*c*b^-2, a*c^2*a*c^2*a*c^-2,
>  (a*c*a^-1*c*a*b^-1*c^-1)^2, (a*c^-1*a^-1*c^-1*a*c*b)^2,
>  a*c*a^2*c^-2*a^-1*c*a^-2*c*a^2*b*c^-1 >;
>
> print Order(G);
0
>
> n,ct:=ToddCoxeter(G,sub<G|>: CosetLimit:=500000);
> print n;
0
>
> Q:=CosetImage(G,sub<G|a>);
> print Degree(Q),Order(Q); print CompositionFactors(Q);
40320 443520
    G
    |  M22
    1
>
> n,ct:=ToddCoxeter(G,sub<G|>: CosetLimit:=2000000);
> print n;
443520
>
> quit;

Total time: 21.330 seconds, Total memory usage: 87.41MB


//////////////////
// Trivial group //
//////////////////

**** Magma input ****

G<x>:=Group< x | x^67741 = x^84053 = 1 >;
print Order(G);

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:01:51 on mcon007-01 [Seed =
2210443845]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<x>:=Group< x | x^67741 = x^84053 = 1 >;
> print Order(G);
1
>
> quit;
```

```
Total time: 0.299 seconds, Total memory usage: 25.98MB


//////////////////////////
// (2,3,6) Triangle group //
//////////////////////////

**** Magma input ****

G<x,y,z>:=Group< x,y,z | x*y*z, x^2, y^3, z^6 >;

G:=Rewrite(G,G); print G;

H:=sub<G| x^-1*y^-1*x*y, x*y*x^-1*y^-1 >;
print Index(G,H); CosetTable(G,H);
f:=CosetAction(G,H); P:=Image(f);  print P;

print Order(G);

K:=Rewrite(G,H); print K;
print AbelianQuotientInvariants(K);

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:02:33 on mcon007-01 [Seed =
2581244329]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<x,y,z>:=Group< x,y,z | x*y*z, x^2, y^3, z^6 >;
>
> G:=Rewrite(G,G); print G;
Finitely presented group G on 2 generators
Relations
    G.1^3 = Id(G)
    (G.2^-1 * G.1^-1)^2 = Id(G)
    G.2^6 = Id(G)
>
> H:=sub<G| x^-1*y^-1*x*y, x*y*x^-1*y^-1 >;
> print Index(G,H); CosetTable(G,H);
6
Mapping from: Cartesian Product<{ 1 .. 6 }, GrpFP: G> to { 1 .. 6 }
    $1  $2 -$1 -$2
1.   2   3   4   5
2.   4   6   1   3
3.   6   2   5   1
4.   1   5   2   6
5.   3   1   6   4
6.   5   4   3   2

> f:=CosetAction(G,H); P:=Image(f);  print P;
Permutation group P acting on a set of cardinality 6
    (1, 2, 4)(3, 6, 5)
    (1, 3, 2, 6, 4, 5)
>
> print Order(G);
0
>
> K:=Rewrite(G,H); print K;
Finitely presented group K on 2 generators
Generators as words in group G
    K.1 = (G.2^-1, G.1^-1)
    K.2 = G.2^2 * G.1^-1
Relations
    (K.1^-1, K.2^-1) = Id(K)
> print AbelianQuotientInvariants(K);
[ 0, 0 ]
>
```

```
> quit;

Total time: 2.300 seconds, Total memory usage: 22.57MB


.....................................................................
..
Demonstration 7:  Subgroups of small index in finitely-presented groups
.....................................................................
..


////////////////////////////
// (2,3,6) Triangle group //
////////////////////////////

**** Magma input ****

G<x,y,z>:=Group< x,y,z | x*y*z, x^2, y^3, z^6 >;

L:=LowIndexSubgroups(G,18);
print #L;

L:=LowIndexSubgroups(G,180);
print #L;

for i in [1..#L] do Q:=CosetImage(G,L[i]);
  if Index(G,L[i]) in [15,16,17,18] then print Degree(Q), Order(Q);
  end if; end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:03:05 on mcon007-01 [Seed =
2631904396]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<x,y,z>:=Group< x,y,z | x*y*z, x^2, y^3, z^6 >;
>
> L:=LowIndexSubgroups(G,18);
> print #L;
43
>
> L:=LowIndexSubgroups(G,180);
> print #L;
2234
>
> for i in [1..#L] do Q:=CosetImage(G,L[i]);
for>   if Index(G,L[i]) in [15,16,17,18] then print Degree(Q), Order(Q);
for|if>   end if; end for;
15 150
15 150
16 96
18 18
18 54
18 54
18 54
18 216
18 216
18 216
18 216
18 72
18 72
18 216
18 216
>
> quit;

Total time: 1.899 seconds, Total memory usage: 15.36MB
```

```
////////////////////////////////////
// Extended (2,3,7) triangle group //
////////////////////////////////////

**** Magma input ****

G<a,b,c>:=Group< a,b,c | a^2, b^2, c^2, (a*b)^2, (b*c)^3, (a*c)^7 >;

L:=LowIndexSubgroups(G,15);
print #L;

for i in [1..#L] do Q:=CosetImage(G,L[i]);
  print Degree(Q), Order(Q), FactoredOrder(Q);
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:03:44 on mcon007-01 [Seed =
4004372135]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<a,b,c>:=Group< a,b,c | a^2, b^2, c^2, (a*b)^2, (b*c)^3, (a*c)^7 >;
>
> L:=LowIndexSubgroups(G,15);
> print #L;
9
>
> for i in [1..#L] do Q:=CosetImage(G,L[i]);
for>   print Degree(Q), Order(Q), FactoredOrder(Q);
for>   end for;
1 1 []
2 2 [ <2, 1> ]
8 336 [ <2, 4>, <3, 1>, <7, 1> ]
9 504 [ <2, 3>, <3, 2>, <7, 1> ]
14 2184 [ <2, 3>, <3, 1>, <7, 1>, <13, 1> ]
14 2184 [ <2, 3>, <3, 1>, <7, 1>, <13, 1> ]
14 336 [ <2, 4>, <3, 1>, <7, 1> ]
14 1092 [ <2, 2>, <3, 1>, <7, 1>, <13, 1> ]
15 653837184000 [ <2, 10>, <3, 6>, <5, 3>, <7, 2>, <11, 1>, <13, 1> ]
>
> quit;

Total time: 0.279 seconds, Total memory usage: 9.20MB


////////////////////////////////////
// Modular group C2 * C3 = PSL(2,Z) //
////////////////////////////////////

**** Magma input ****

G<x,y>:=Group< x,y | x^2, y^3 >;

for n in [1..10] do L:=LowIndexSubgroups(G,n);
  print "Index up to",n,"  Number of classes of subgroups =",#L;
  end for;

for n in [11..20] do L:=LowIndexSubgroups(G,n);
  print "Index up to",n,"  Number of classes of subgroups =",#L;
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:04:19 on mcon007-01 [Seed =
3852916040]
Type ? for help.  Type <Ctrl>-D to quit.
```

```
>
> G<x,y>:=Group< x,y | x^2, y^3 >;
>
> for n in [1..10] do L:=LowIndexSubgroups(G,n);
for>   print "Index up to",n,"  Number of classes of subgroups =",#L;
for>   end for;
Index up to 1   Number of classes of subgroups = 1
Index up to 2   Number of classes of subgroups = 2
Index up to 3   Number of classes of subgroups = 4
Index up to 4   Number of classes of subgroups = 6
Index up to 5   Number of classes of subgroups = 7
Index up to 6   Number of classes of subgroups = 15
Index up to 7   Number of classes of subgroups = 21
Index up to 8   Number of classes of subgroups = 28
Index up to 9   Number of classes of subgroups = 42
Index up to 10   Number of classes of subgroups = 69
>
> for n in [11..20] do L:=LowIndexSubgroups(G,n);
for>   print "Index up to",n,"  Number of classes of subgroups =",#L;
for>   end for;
Index up to 11   Number of classes of subgroups = 95
Index up to 12   Number of classes of subgroups = 175
Index up to 13   Number of classes of subgroups = 308
Index up to 14   Number of classes of subgroups = 478
Index up to 15   Number of classes of subgroups = 826
Index up to 16   Number of classes of subgroups = 1591
Index up to 17   Number of classes of subgroups = 2593
Index up to 18   Number of classes of subgroups = 4769
Index up to 19   Number of classes of subgroups = 9451
Index up to 20   Number of classes of subgroups = 16382
>
> quit;

Total time: 7.239 seconds, Total memory usage: 42.54MB


.....................................................................
..
Demonstration 8:  Normal subgroups of small index
.....................................................................
..


////////////////////////////////////
// Modular group C2 * C3 = PSL(2,Z) //
////////////////////////////////////

**** Magma input ****

G<x,y>:=Group< x,y | x^2, y^3 >;

n:=20;
L:=LowIndexNormalSubgroups(G,n);
print "Index up to",n,"  Number of normal subgroups =",#L;

for n in [20*t : t in [1..15]] do
  L:=LowIndexNormalSubgroups(G,n);
  print "Index up to",n,"  Number of normal subgroups =",#L;
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:05:00 on mcon007-01 [Seed =
4173056125]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G<x,y>:=Group< x,y | x^2, y^3 >;
>
```

```
> n:=20;
> L:=LowIndexNormalSubgroups(G,n);
> print "Index up to",n,"  Number of normal subgroups =",#L;
Index up to 20   Number of normal subgroups = 7
>
> for n in [20*t : t in [1..15]] do
for>   L:=LowIndexNormalSubgroups(G,n);
for>   print "Index up to",n,"  Number of normal subgroups =",#L;
for>   end for;
Index up to 20   Number of normal subgroups = 7
Index up to 40   Number of normal subgroups = 9
Index up to 60   Number of normal subgroups = 15
Index up to 80   Number of normal subgroups = 19
Index up to 100   Number of normal subgroups = 21
Index up to 120   Number of normal subgroups = 24
Index up to 140   Number of normal subgroups = 26
Index up to 160   Number of normal subgroups = 30
Index up to 180   Number of normal subgroups = 36
Index up to 200   Number of normal subgroups = 40
Index up to 220   Number of normal subgroups = 42
Index up to 240   Number of normal subgroups = 47
Index up to 260   Number of normal subgroups = 49
Index up to 280   Number of normal subgroups = 49
Index up to 300   Number of normal subgroups = 56
>
> quit;

Total time: 5.980 seconds, Total memory usage: 45.63MB


////////////////////////////////////
// The Djokovic-Miller amalgam G_5 //
////////////////////////////////////

**** Magma input ****

G5<h,p,q,r,s,a>:=Group<h,p,q,r,s,a | h^3,p^2,q^2,r^2,s^2,a^2,
(p,q),(p,r),(p,s),(q,r),(q,s),p*q*(r*s)^2,
(h,p),h^-1*q*h*r,h^-1*r*h*p*q*r,(s*h)^2, a*p*a*q,a*r*a*s>;

n:=4800;
L:=LowIndexNormalSubgroups(G5,n);

print "Index up to",n,"  Number of normal subgroups =",#L;
for i in [1..#L] do Q:=CosetImage(G5,L[i]`Group);
  print i,Order(Q),FactoredOrder(Q); end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:06:20 on mcon007-01 [Seed =
4071735847]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G5<h,p,q,r,s,a>:=Group<h,p,q,r,s,a | h^3,p^2,q^2,r^2,s^2,a^2,
> (p,q),(p,r),(p,s),(q,r),(q,s),p*q*(r*s)^2,
> (h,p),h^-1*q*h*r,h^-1*r*h*p*q*r,(s*h)^2, a*p*a*q,a*r*a*s>;
>
> n:=4800;
> L:=LowIndexNormalSubgroups(G5,n);
>
> print "Index up to",n,"  Number of normal subgroups =",#L;
Index up to 4800   Number of normal subgroups = 7
> for i in [1..#L] do Q:=CosetImage(G5,L[i]`Group);
for>   print i,Order(Q),FactoredOrder(Q); end for;
1 1 []
2 2 [ <2, 1> ]
3 2 [ <2, 1> ]
4 2 [ <2, 1> ]
```

```
5 4 [ <2, 2> ]
6 1440 [ <2, 5>, <3, 2>, <5, 1> ]
7 4320 [ <2, 5>, <3, 3>, <5, 1> ]
>
> quit;

Total time: 2.410 seconds, Total memory usage: 29.88MB


.....................................................................
..
Demonstration 9:  Use of low index normal subgroups to generate regular
maps
.....................................................................
..


////////////////////////////////////////////
// Reflexible regular maps of genus 2 to 15 //
////////////////////////////////////////////

**** Magma input ****

maxgenus:=15;  maxo:=4*maxgenus+2;  ctm:=0;

for p in [2..maxo] do for q in [p..maxo] do if (1/p+1/q lt 1/2) then
  ratio:=(1/2-(1/p+1/q));  lcm:=LCM(2,LCM(p,q));
  maxn:=(Floor(4*(maxgenus-1)/ratio) div lcm)*lcm;
  if maxn gt 0 then
    F<a,b,c>:=Group<a,b,c| a^2,b^2,c^2, (a*c)^2,(a*b)^p,(b*c)^q >;
    L:=LowIndexNormalSubgroups(F,maxn);
    // print p,q,":",#L,"normal subgroups of index up to",maxn;
    for i in [1..#L] do f:=CosetAction(F,L[i]`Group); Q:=Image(f);
      if [Order(f(x)) : x in [a,b,c,a*c,a*b,b*c]] eq [2,2,2,2,p,q] then
      if Index(Q,sub<Q|f(a*b),f(b*c),f(a*c)>) eq 2 then
        n:=Order(Q); chi:=(n div (2*p)) - (n div 4) + (n div (2*q));
        g:=(2-chi) div 2;  ctm:=ctm+1;
        print "Genus",g,"reflexible regular map of type",[p,q];
        end if;end if;
      end for;
    end if;
  end if; end for;end for;

print "Total number of maps found =",ctm;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:06:20 on mcon007-01 [Seed =
4071735847]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G5<h,p,q,r,s,a>:=Group<h,p,q,r,s,a | h^3,p^2,q^2,r^2,s^2,a^2,
> (p,q),(p,r),(p,s),(q,r),(q,s),p*q*(r*s)^2,
> (h,p),h^-1*q*h*r,h^-1*r*h*p*q*r,(s*h)^2, a*p*a*q,a*r*a*s>;
>
> n:=4800;
> L:=LowIndexNormalSubgroups(G5,n);
>
> print "Index up to",n,"  Number of normal subgroups =",#L;
Index up to 4800   Number of normal subgroups = 7
> for i in [1..#L] do Q:=CosetImage(G5,L[i]`Group);
for>   print i,Order(Q),FactoredOrder(Q); end for;
1 1 []
2 2 [ <2, 1> ]
3 2 [ <2, 1> ]
4 2 [ <2, 1> ]
5 4 [ <2, 2> ]
6 1440 [ <2, 5>, <3, 2>, <5, 1> ]
```

```
7 4320 [ <2, 5>, <3, 3>, <5, 1> ]
>
> quit;

Total time: 2.410 seconds, Total memory usage: 29.88MB
mcon007-01:~ mcon007$ magma
Magma V2.15-15    Fri Jul  1 2011 20:07:13 on mcon007-01 [Seed =
3381266129]
Type ? for help.  Type <Ctrl>-D to quit.
>
> maxgenus:=15;  maxo:=4*maxgenus+2;  ctm:=0;
>
> for p in [2..maxo] do for q in [p..maxo] do if (1/p+1/q lt 1/2) then
for|for|if>   ratio:=(1/2-(1/p+1/q));  lcm:=LCM(2,LCM(p,q));
for|for|if>   maxn:=(Floor(4*(maxgenus-1)/ratio) div lcm)*lcm;
for|for|if>   if maxn gt 0 then
for|for|if|if>     F<a,b,c>:=Group<a,b,c| a^2,b^2,c^2, (a*c)^2,(a*b)
^p,(b*c)^q\
 >;
for|for|if|if>     L:=LowIndexNormalSubgroups(F,maxn);
for|for|if|if>     // print p,q,":",#L,"normal subgroups of index up
to",maxn;
for|for|if|if>     for i in [1..#L] do f:=CosetAction(F,L[i]`Group); Q:=
Image(\
f);
for|for|if|if|for>       if [Order(f(x)) : x in [a,b,c,a*c,a*b,b*c]] eq
[2,2,2\
,2,p,q] then
for|for|if|if|for|if>       if Index(Q,sub<Q|f(a*b),f(b*c),f(a*c)>) eq 2
then
for|for|if|if|for|if|if>         n:=Order(Q); chi:=(n div (2*p)) - (n div
4) +\
 (n div (2*q));
for|for|if|if|for|if|if>         g:=(2-chi) div 2;  ctm:=ctm+1;
for|for|if|if|for|if|if>         print "Genus",g,"reflexible regular map of
ty\
pe",[p,q];
for|for|if|if|for|if|if>         end if;end if;
for|for|if|if|for>       end for;
for|for|if|if>     end if;
for|for|if>   end if; end for;end for;
Genus 3 reflexible regular map of type [ 3, 7 ]
Genus 7 reflexible regular map of type [ 3, 7 ]
Genus 14 reflexible regular map of type [ 3, 7 ]
Genus 14 reflexible regular map of type [ 3, 7 ]
Genus 14 reflexible regular map of type [ 3, 7 ]
Genus 2 reflexible regular map of type [ 3, 8 ]
Genus 3 reflexible regular map of type [ 3, 8 ]
Genus 5 reflexible regular map of type [ 3, 8 ]
Genus 8 reflexible regular map of type [ 3, 8 ]
Genus 8 reflexible regular map of type [ 3, 8 ]
Genus 10 reflexible regular map of type [ 3, 9 ]
Genus 15 reflexible regular map of type [ 3, 9 ]
Genus 5 reflexible regular map of type [ 3, 10 ]
Genus 6 reflexible regular map of type [ 3, 10 ]
Genus 13 reflexible regular map of type [ 3, 10 ]
Genus 3 reflexible regular map of type [ 3, 12 ]
Genus 4 reflexible regular map of type [ 3, 12 ]
Genus 7 reflexible regular map of type [ 3, 12 ]
Genus 9 reflexible regular map of type [ 3, 12 ]
Genus 10 reflexible regular map of type [ 3, 12 ]
Genus 13 reflexible regular map of type [ 3, 12 ]
Genus 15 reflexible regular map of type [ 3, 14 ]
Genus 10 reflexible regular map of type [ 3, 15 ]
Genus 10 reflexible regular map of type [ 3, 18 ]
Genus 15 reflexible regular map of type [ 3, 20 ]
Genus 10 reflexible regular map of type [ 3, 24 ]
Genus 4 reflexible regular map of type [ 4, 5 ]
```

Genus 5 reflexible regular map of type [4, 5]
Genus 9 reflexible regular map of type [4, 5]
Genus 10 reflexible regular map of type [4, 5]
Genus 2 reflexible regular map of type [4, 6]
Genus 3 reflexible regular map of type [4, 6]
Genus 4 reflexible regular map of type [4, 6]
Genus 5 reflexible regular map of type [4, 6]
Genus 6 reflexible regular map of type [4, 6]
Genus 9 reflexible regular map of type [4, 6]
Genus 9 reflexible regular map of type [4, 6]
Genus 10 reflexible regular map of type [4, 6]
Genus 10 reflexible regular map of type [4, 6]
Genus 11 reflexible regular map of type [4, 6]
Genus 15 reflexible regular map of type [4, 6]
Genus 10 reflexible regular map of type [4, 7]
Genus 2 reflexible regular map of type [4, 8]
Genus 3 reflexible regular map of type [4, 8]
Genus 3 reflexible regular map of type [4, 8]
Genus 5 reflexible regular map of type [4, 8]
Genus 5 reflexible regular map of type [4, 8]
Genus 9 reflexible regular map of type [4, 8]
Genus 9 reflexible regular map of type [4, 8]
Genus 9 reflexible regular map of type [4, 8]
Genus 9 reflexible regular map of type [4, 8]
Genus 6 reflexible regular map of type [4, 9]
Genus 4 reflexible regular map of type [4, 10]
Genus 3 reflexible regular map of type [4, 12]
Genus 5 reflexible regular map of type [4, 12]
Genus 9 reflexible regular map of type [4, 12]
Genus 9 reflexible regular map of type [4, 12]
Genus 9 reflexible regular map of type [4, 12]
Genus 10 reflexible regular map of type [4, 12]
Genus 13 reflexible regular map of type [4, 12]
Genus 6 reflexible regular map of type [4, 14]
Genus 12 reflexible regular map of type [4, 15]
Genus 4 reflexible regular map of type [4, 16]
Genus 7 reflexible regular map of type [4, 16]
Genus 7 reflexible regular map of type [4, 16]
Genus 13 reflexible regular map of type [4, 16]
Genus 13 reflexible regular map of type [4, 16]
Genus 8 reflexible regular map of type [4, 18]
Genus 15 reflexible regular map of type [4, 18]
Genus 5 reflexible regular map of type [4, 20]
Genus 9 reflexible regular map of type [4, 20]
Genus 10 reflexible regular map of type [4, 22]
Genus 6 reflexible regular map of type [4, 24]
Genus 11 reflexible regular map of type [4, 24]
Genus 11 reflexible regular map of type [4, 24]
Genus 12 reflexible regular map of type [4, 26]
Genus 7 reflexible regular map of type [4, 28]
Genus 13 reflexible regular map of type [4, 28]
Genus 14 reflexible regular map of type [4, 30]
Genus 8 reflexible regular map of type [4, 32]
Genus 15 reflexible regular map of type [4, 32]
Genus 15 reflexible regular map of type [4, 32]
Genus 9 reflexible regular map of type [4, 36]
Genus 10 reflexible regular map of type [4, 40]
Genus 11 reflexible regular map of type [4, 44]
Genus 12 reflexible regular map of type [4, 48]
Genus 13 reflexible regular map of type [4, 52]
Genus 14 reflexible regular map of type [4, 56]
Genus 15 reflexible regular map of type [4, 60]
Genus 4 reflexible regular map of type [5, 5]
Genus 5 reflexible regular map of type [5, 5]
Genus 9 reflexible regular map of type [5, 5]
Genus 9 reflexible regular map of type [5, 6]
Genus 9 reflexible regular map of type [5, 6]
Genus 2 reflexible regular map of type [5, 10]

Genus 6 reflexible regular map of type [5, 10]
Genus 13 reflexible regular map of type [5, 10]
Genus 2 reflexible regular map of type [6, 6]
Genus 3 reflexible regular map of type [6, 6]
Genus 4 reflexible regular map of type [6, 6]
Genus 4 reflexible regular map of type [6, 6]
Genus 4 reflexible regular map of type [6, 6]
Genus 5 reflexible regular map of type [6, 6]
Genus 9 reflexible regular map of type [6, 6]
Genus 9 reflexible regular map of type [6, 6]
Genus 10 reflexible regular map of type [6, 6]
Genus 10 reflexible regular map of type [6, 6]
Genus 10 reflexible regular map of type [6, 6]
Genus 10 reflexible regular map of type [6, 6]
Genus 11 reflexible regular map of type [6, 6]
Genus 13 reflexible regular map of type [6, 6]
Genus 13 reflexible regular map of type [6, 6]
Genus 6 reflexible regular map of type [6, 8]
Genus 6 reflexible regular map of type [6, 8]
Genus 11 reflexible regular map of type [6, 8]
Genus 11 reflexible regular map of type [6, 8]
Genus 7 reflexible regular map of type [6, 9]
Genus 8 reflexible regular map of type [6, 10]
Genus 15 reflexible regular map of type [6, 10]
Genus 4 reflexible regular map of type [6, 12]
Genus 7 reflexible regular map of type [6, 12]
Genus 10 reflexible regular map of type [6, 12]
Genus 10 reflexible regular map of type [6, 12]
Genus 10 reflexible regular map of type [6, 12]
Genus 13 reflexible regular map of type [6, 12]
Genus 13 reflexible regular map of type [6, 12]
Genus 12 reflexible regular map of type [6, 14]
Genus 5 reflexible regular map of type [6, 15]
Genus 13 reflexible regular map of type [6, 15]
Genus 14 reflexible regular map of type [6, 16]
Genus 7 reflexible regular map of type [6, 21]
Genus 8 reflexible regular map of type [6, 24]
Genus 10 reflexible regular map of type [6, 30]
Genus 11 reflexible regular map of type [6, 33]
Genus 13 reflexible regular map of type [6, 39]
Genus 14 reflexible regular map of type [6, 42]
Genus 3 reflexible regular map of type [7, 14]
Genus 15 reflexible regular map of type [7, 14]
Genus 2 reflexible regular map of type [8, 8]
Genus 3 reflexible regular map of type [8, 8]
Genus 3 reflexible regular map of type [8, 8]
Genus 5 reflexible regular map of type [8, 8]
Genus 5 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 9 reflexible regular map of type [8, 8]
Genus 12 reflexible regular map of type [8, 10]
Genus 8 reflexible regular map of type [8, 12]
Genus 15 reflexible regular map of type [8, 12]
Genus 15 reflexible regular map of type [8, 12]
Genus 15 reflexible regular map of type [8, 12]
Genus 15 reflexible regular map of type [8, 12]
Genus 11 reflexible regular map of type [8, 16]
Genus 11 reflexible regular map of type [8, 16]
Genus 14 reflexible regular map of type [8, 20]
Genus 9 reflexible regular map of type [8, 24]
Genus 9 reflexible regular map of type [8, 24]
Genus 15 reflexible regular map of type [8, 40]
Genus 15 reflexible regular map of type [8, 40]
Genus 6 reflexible regular map of type [9, 9]

```
Genus 4 reflexible regular map of type [ 9, 18 ]
Genus 10 reflexible regular map of type [ 9, 18 ]
Genus 13 reflexible regular map of type [ 9, 18 ]
Genus 4 reflexible regular map of type [ 10, 10 ]
Genus 6 reflexible regular map of type [ 10, 15 ]
Genus 8 reflexible regular map of type [ 10, 20 ]
Genus 12 reflexible regular map of type [ 10, 30 ]
Genus 14 reflexible regular map of type [ 10, 35 ]
Genus 5 reflexible regular map of type [ 11, 22 ]
Genus 3 reflexible regular map of type [ 12, 12 ]
Genus 5 reflexible regular map of type [ 12, 12 ]
Genus 9 reflexible regular map of type [ 12, 12 ]
Genus 9 reflexible regular map of type [ 12, 12 ]
Genus 9 reflexible regular map of type [ 12, 12 ]
Genus 13 reflexible regular map of type [ 12, 12 ]
Genus 13 reflexible regular map of type [ 12, 12 ]
Genus 13 reflexible regular map of type [ 12, 12 ]
Genus 10 reflexible regular map of type [ 12, 24 ]
Genus 6 reflexible regular map of type [ 13, 26 ]
Genus 6 reflexible regular map of type [ 14, 14 ]
Genus 9 reflexible regular map of type [ 14, 21 ]
Genus 12 reflexible regular map of type [ 14, 28 ]
Genus 15 reflexible regular map of type [ 14, 35 ]
Genus 12 reflexible regular map of type [ 15, 15 ]
Genus 7 reflexible regular map of type [ 15, 30 ]
Genus 4 reflexible regular map of type [ 16, 16 ]
Genus 7 reflexible regular map of type [ 16, 16 ]
Genus 7 reflexible regular map of type [ 16, 16 ]
Genus 13 reflexible regular map of type [ 16, 16 ]
Genus 13 reflexible regular map of type [ 16, 16 ]
Genus 13 reflexible regular map of type [ 16, 16 ]
Genus 13 reflexible regular map of type [ 16, 16 ]
Genus 8 reflexible regular map of type [ 17, 34 ]
Genus 8 reflexible regular map of type [ 18, 18 ]
Genus 15 reflexible regular map of type [ 18, 18 ]
Genus 9 reflexible regular map of type [ 19, 38 ]
Genus 5 reflexible regular map of type [ 20, 20 ]
Genus 9 reflexible regular map of type [ 20, 20 ]
Genus 10 reflexible regular map of type [ 21, 42 ]
Genus 10 reflexible regular map of type [ 22, 22 ]
Genus 15 reflexible regular map of type [ 22, 33 ]
Genus 11 reflexible regular map of type [ 23, 46 ]
Genus 6 reflexible regular map of type [ 24, 24 ]
Genus 11 reflexible regular map of type [ 24, 24 ]
Genus 11 reflexible regular map of type [ 24, 24 ]
Genus 12 reflexible regular map of type [ 25, 50 ]
Genus 12 reflexible regular map of type [ 26, 26 ]
Genus 13 reflexible regular map of type [ 27, 54 ]
Genus 7 reflexible regular map of type [ 28, 28 ]
Genus 13 reflexible regular map of type [ 28, 28 ]
Genus 14 reflexible regular map of type [ 29, 58 ]
Genus 14 reflexible regular map of type [ 30, 30 ]
Genus 15 reflexible regular map of type [ 31, 62 ]
Genus 8 reflexible regular map of type [ 32, 32 ]
Genus 15 reflexible regular map of type [ 32, 32 ]
Genus 15 reflexible regular map of type [ 32, 32 ]
Genus 9 reflexible regular map of type [ 36, 36 ]
Genus 10 reflexible regular map of type [ 40, 40 ]
Genus 11 reflexible regular map of type [ 44, 44 ]
Genus 12 reflexible regular map of type [ 48, 48 ]
Genus 13 reflexible regular map of type [ 52, 52 ]
Genus 14 reflexible regular map of type [ 56, 56 ]
Genus 15 reflexible regular map of type [ 60, 60 ]
>
> print "Total number of maps found =",ctm;
Total number of maps found = 226
>
> quit;
```

```
Total time: 243.710 seconds, Total memory usage: 47.22MB


.....................................................................
..
Demonstration 10:  Use of low index normal subgroups to generate graphs
.....................................................................
..


////////////////////////////////////
// Cayley graphs of order 20 to 24 //
////////////////////////////////////

**** Magma input ****

F<a,b,c>:=Group<a,b,c|a^2,b^2,c^2>;
L:=LowIndexNormalSubgroups(F,24);

graphs:=[];print "";
for t in [1..#L] do
  rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
  if (Order(Q) in [20..24]) and (#{1,1^f(a),1^f(b),1^f(c)} eq 4) then
    edges:={{1,1^f(u)}^g : u in [a,b,c], g in Q};
    cgs:=Graph<Degree(Q)|edges>;  okm:=false;
    for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=true;break gr;end
    if;end for;
    if not(okm) then Append(~graphs,cgs);
      print #graphs,": Cayley graph of order",Order(cgs),

      "valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs),"diameter",D
      iameter(cgs);
      end if;
    end if;
  end for;

F<x,y>:=Group<x,y|x^2>;
L:=LowIndexNormalSubgroups(F,24);

graphs2:=[];print "";
for t in [1..#L] do
  rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
  if (Order(Q) in [20..24]) and (#{1,1^f(x),1^f(y),1^f(y^-1)} eq 4) then
    edges:={{1,1^f(u)}^g : u in [x,y,y^-1], g in Q};
    cgs:=Graph<Degree(Q)|edges>;  okm:=false;
    for gr in graphs2 do if IsIsomorphic(cgs,gr) then okm:=true;break
    gr;end if;end for;
    if not(okm) then Append(~graphs2,cgs);
      print #graphs2,": Cayley graph of order",Order(cgs),

      "valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs),"diameter",D
      iameter(cgs);
      end if;
    end if;
  end for;

for i in [1..#graphs] do for j in [1..#graphs2] do
  if IsIsomorphic(graphs[i],graphs2[j]) then
    print "F1 graph",i,"of order",Order(graphs[i]),
     "isomorphic to F2 graph",j,"of order",Order(graphs2[j]);
    end if;
  end for;end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:08:18 on mcon007-01 [Seed =
3229814683]
```

```
Type ? for help.  Type <Ctrl>-D to quit.
>
> F<a,b,c>:=Group<a,b,c|a^2,b^2,c^2>;
> L:=LowIndexNormalSubgroups(F,24);
>
> graphs:=[];print "";

> for t in [1..#L] do
for>   rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
for>   if (Order(Q) in [20..24]) and (#{1,1^f(a),1^f(b),1^f(c)} eq 4) then
for|if>     edges:={{1,1^f(u)}^g : u in [a,b,c], g in Q};
for|if>     cgs:=Graph<Degree(Q)|edges>;  okm:=false;
for|if>     for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=
true;break g\
r;end if;end for;
for|if>     if not(okm) then Append(~graphs,cgs);
for|if|if>       print #graphs,": Cayley graph of order",Order(cgs),
for|if|if|print>
"valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs\
),"diameter",Diameter(cgs);
for|if|if>       end if;
for|if>     end if;
for>   end for;
1 : Cayley graph of order 20 valency 3 girth 4 diameter 6
2 : Cayley graph of order 20 valency 3 girth 4 diameter 5
3 : Cayley graph of order 20 valency 3 girth 6 diameter 4
4 : Cayley graph of order 20 valency 3 girth 4 diameter 5
5 : Cayley graph of order 22 valency 3 girth 4 diameter 6
6 : Cayley graph of order 22 valency 3 girth 6 diameter 5
7 : Cayley graph of order 24 valency 3 girth 4 diameter 6
8 : Cayley graph of order 24 valency 3 girth 4 diameter 6
9 : Cayley graph of order 24 valency 3 girth 6 diameter 5
10 : Cayley graph of order 24 valency 3 girth 6 diameter 5
11 : Cayley graph of order 24 valency 3 girth 6 diameter 4
12 : Cayley graph of order 24 valency 3 girth 6 diameter 5
13 : Cayley graph of order 24 valency 3 girth 4 diameter 7
14 : Cayley graph of order 24 valency 3 girth 6 diameter 4
15 : Cayley graph of order 24 valency 3 girth 4 diameter 4
16 : Cayley graph of order 24 valency 3 girth 4 diameter 6
>
> F<x,y>:=Group<x,y|x^2>;
> L:=LowIndexNormalSubgroups(F,24);
>
> graphs2:=[];print "";

> for t in [1..#L] do
for>   rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
for>   if (Order(Q) in [20..24]) and (#{1,1^f(x),1^f(y),1^f(y^-1)} eq 4)
then
for|if>     edges:={{1,1^f(u)}^g : u in [x,y,y^-1], g in Q};
for|if>     cgs:=Graph<Degree(Q)|edges>;  okm:=false;
for|if>     for gr in graphs2 do if IsIsomorphic(cgs,gr) then okm:=
true;break \
gr;end if;end for;
for|if>     if not(okm) then Append(~graphs2,cgs);
for|if|if>       print #graphs2,": Cayley graph of order",Order(cgs),
for|if|if|print>
"valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs\
),"diameter",Diameter(cgs);
for|if|if>       end if;
for|if>     end if;
for>   end for;
1 : Cayley graph of order 20 valency 3 girth 4 diameter 4
2 : Cayley graph of order 20 valency 3 girth 4 diameter 5
3 : Cayley graph of order 20 valency 3 girth 4 diameter 6
4 : Cayley graph of order 22 valency 3 girth 4 diameter 6
5 : Cayley graph of order 22 valency 3 girth 4 diameter 6
6 : Cayley graph of order 24 valency 3 girth 3 diameter 6
```

```
7 : Cayley graph of order 24 valency 3 girth 6 diameter 4
8 : Cayley graph of order 24 valency 3 girth 6 diameter 4
9 : Cayley graph of order 24 valency 3 girth 4 diameter 7
10 : Cayley graph of order 24 valency 3 girth 6 diameter 5
11 : Cayley graph of order 24 valency 3 girth 4 diameter 6
12 : Cayley graph of order 24 valency 3 girth 4 diameter 6
13 : Cayley graph of order 24 valency 3 girth 4 diameter 6
>
> for i in [1..#graphs] do for j in [1..#graphs2] do
for|for>   if IsIsomorphic(graphs[i],graphs2[j]) then
for|for|if>     print "F1 graph",i,"of order",Order(graphs[i]),
for|for|if|print>      "isomorphic to F2 graph",j,"of
order",Order(graphs2[j])\
;
for|for|if>     end if;
for|for>   end for;end for;
F1 graph 1 of order 20 isomorphic to F2 graph 3 of order 20
F1 graph 4 of order 20 isomorphic to F2 graph 2 of order 20
F1 graph 5 of order 22 isomorphic to F2 graph 4 of order 22
F1 graph 7 of order 24 isomorphic to F2 graph 12 of order 24
F1 graph 8 of order 24 isomorphic to F2 graph 13 of order 24
F1 graph 11 of order 24 isomorphic to F2 graph 7 of order 24
F1 graph 12 of order 24 isomorphic to F2 graph 10 of order 24
F1 graph 13 of order 24 isomorphic to F2 graph 9 of order 24
F1 graph 14 of order 24 isomorphic to F2 graph 8 of order 24
F1 graph 16 of order 24 isomorphic to F2 graph 11 of order 24
>
> quit;

Total time: 5.290 seconds, Total memory usage: 45.96MB


////////////////////////////////
// Cayley graphs of order 128 //
////////////////////////////////

**** Magma input ****

graphs:=[];

F<x,y>:=Group<x,y|x^2>;
L:=LowIndexNormalSubgroups(F,128);
print "";

for t in [1..#L] do
  rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
  if (Order(Q) eq 128) and (#{1,1^f(x),1^f(y),1^f(y^-1)} eq 4) then
    edges:={{1,1^f(u)}^g : u in [x,y,y^-1], g in Q};
    cgs:=Graph<Degree(Q)|edges>;  okm:=false;
    for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=true;break gr;end
    if;end for;
    if not(okm) then Append(~graphs,cgs);
      print #graphs,": Cayley graph of order",Order(cgs),

      "valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs),"diameter",D
      iameter(cgs);
      end if;
    end if;
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:10:03 on mcon007-01 [Seed =
3549436961]
Type ? for help.  Type <Ctrl>-D to quit.
>
> graphs:=[];
>
```

```
> F<x,y>:=Group<x,y|x^2>;
> L:=LowIndexNormalSubgroups(F,128);
> print "";

>
> for t in [1..#L] do
for>   rep:=L[t]`Group;  f:=CosetAction(F,rep);  Q:=Image(f);
for>   if (Order(Q) eq 128) and (#{1,1^f(x),1^f(y),1^f(y^-1)} eq 4) then
for|if>     edges:={{1,1^f(u)}^g : u in [x,y,y^-1], g in Q};
for|if>     cgs:=Graph<Degree(Q)|edges>;  okm:=false;
for|if>     for gr in graphs do if IsIsomorphic(cgs,gr) then okm:=
true;break g\
r;end if;end for;
for|if>     if not(okm) then Append(~graphs,cgs);
for|if|if>       print #graphs,": Cayley graph of order",Order(cgs),
for|if|if|print>
"valency",Degree(Rep(VertexSet(cgs))),"girth",Girth(cgs\
),"diameter",Diameter(cgs);
for|if|if>       end if;
for|if>     end if;
for>   end for;
1 : Cayley graph of order 128 valency 3 girth 4 diameter 32
2 : Cayley graph of order 128 valency 3 girth 4 diameter 33
3 : Cayley graph of order 128 valency 3 girth 6 diameter 17
4 : Cayley graph of order 128 valency 3 girth 6 diameter 18
5 : Cayley graph of order 128 valency 3 girth 8 diameter 10
6 : Cayley graph of order 128 valency 3 girth 6 diameter 10
7 : Cayley graph of order 128 valency 3 girth 8 diameter 10
8 : Cayley graph of order 128 valency 3 girth 8 diameter 10
9 : Cayley graph of order 128 valency 3 girth 8 diameter 10
10 : Cayley graph of order 128 valency 3 girth 6 diameter 12
11 : Cayley graph of order 128 valency 3 girth 10 diameter 8
12 : Cayley graph of order 128 valency 3 girth 8 diameter 9
13 : Cayley graph of order 128 valency 3 girth 10 diameter 8
14 : Cayley graph of order 128 valency 3 girth 8 diameter 10
15 : Cayley graph of order 128 valency 3 girth 8 diameter 10
16 : Cayley graph of order 128 valency 3 girth 8 diameter 10
17 : Cayley graph of order 128 valency 3 girth 10 diameter 8
18 : Cayley graph of order 128 valency 3 girth 8 diameter 10
19 : Cayley graph of order 128 valency 3 girth 8 diameter 8
20 : Cayley graph of order 128 valency 3 girth 10 diameter 9
21 : Cayley graph of order 128 valency 3 girth 6 diameter 11
22 : Cayley graph of order 128 valency 3 girth 10 diameter 9
23 : Cayley graph of order 128 valency 3 girth 6 diameter 16
24 : Cayley graph of order 128 valency 3 girth 8 diameter 9
25 : Cayley graph of order 128 valency 3 girth 10 diameter 9
26 : Cayley graph of order 128 valency 3 girth 8 diameter 9
27 : Cayley graph of order 128 valency 3 girth 4 diameter 10
28 : Cayley graph of order 128 valency 3 girth 4 diameter 16
29 : Cayley graph of order 128 valency 3 girth 8 diameter 8
30 : Cayley graph of order 128 valency 3 girth 8 diameter 9
31 : Cayley graph of order 128 valency 3 girth 4 diameter 12
32 : Cayley graph of order 128 valency 3 girth 4 diameter 9
33 : Cayley graph of order 128 valency 3 girth 8 diameter 9
34 : Cayley graph of order 128 valency 3 girth 6 diameter 16
35 : Cayley graph of order 128 valency 3 girth 4 diameter 32
>
> quit;

Total time: 58.259 seconds, Total memory usage: 45.65MB

// Note: The analogous computation using the database of all small groups
// takes about half an hour


////////////////////////////////////
// 5-arc-transitive 3-valent graphs //
////////////////////////////////////
```

```
**** Magma input ****

G5<h,p,q,r,s,a>:=Group<h,p,q,r,s,a | h^3,p^2,q^2,r^2,s^2,a^2,
(p,q),(p,r),(p,s),(q,r),(q,s),p*q*(r*s)^2,
(h,p),h^-1*q*h*r,h^-1*r*h*p*q*r,(s*h)^2, a*p*a*q,a*r*a*s>;

L:=LowIndexNormalSubgroups(G5,4800);

graphs:=[];print "";
for t in [1..#L] do
  rep:=L[t]`Group;  f:=CosetAction(G5,rep);  Q:=Image(f);
  if (Order(Q) gt 100) then
    ff:=CosetAction(Q,sub<Q|f(h),f(p),f(r),f(s)>); QQ:=Image(ff);
    edges:={1,1^ff(f(a))}^QQ;  atg:=Graph<Degree(QQ)|edges>;  okm:=false;
    for gr in graphs do if IsIsomorphic(atg,gr) then okm:=true;break gr;end
    if;end for;
    if not(okm) then Append(~graphs,atg);
      print #graphs,": Symmetric graph of order",Order(atg),

      "valency",Degree(Rep(VertexSet(atg))),"girth",Girth(atg),"diameter",D
      iameter(atg);
      end if;
    end if;
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:12:17 on mcon007-01 [Seed = 694871904]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G5<h,p,q,r,s,a>:=Group<h,p,q,r,s,a | h^3,p^2,q^2,r^2,s^2,a^2,
> (p,q),(p,r),(p,s),(q,r),(q,s),p*q*(r*s)^2,
> (h,p),h^-1*q*h*r,h^-1*r*h*p*q*r,(s*h)^2, a*p*a*q,a*r*a*s>;
>
> L:=LowIndexNormalSubgroups(G5,4800);
>
> graphs:=[];print "";

> for t in [1..#L] do
for>   rep:=L[t]`Group;  f:=CosetAction(G5,rep);  Q:=Image(f);
for>   if (Order(Q) gt 100) then
for|if>     ff:=CosetAction(Q,sub<Q|f(h),f(p),f(r),f(s)>); QQ:=Image(ff);
for|if>     edges:={1,1^ff(f(a))}^QQ;  atg:=Graph<Degree(QQ)|edges>;  okm:=
fal\
se;
for|if>     for gr in graphs do if IsIsomorphic(atg,gr) then okm:=
true;break g\
r;end if;end for;
for|if>     if not(okm) then Append(~graphs,atg);
for|if|if>       print #graphs,": Symmetric graph of order",Order(atg),
for|if|if|print>
"valency",Degree(Rep(VertexSet(atg))),"girth",Girth(atg\
),"diameter",Diameter(atg);
for|if|if>       end if;
for|if>     end if;
for>   end for;
1 : Symmetric graph of order 30 valency 3 girth 8 diameter 4
2 : Symmetric graph of order 90 valency 3 girth 10 diameter 8
>
> quit;

Total time: 2.409 seconds, Total memory usage: 29.88MB


////////////////////////////////////////////////
// 3-arc-transitive covers of the Petersen graph //
////////////////////////////////////////////////
```

```
**** Magma input ****

G3<h,p,q,a>:=Group<h,p,q,a | h^3,p^2,q^2,a^2,(p,q),(h,p),(q*h)^2,a*p*a*q>;

L:=LowIndexSubgroups(G3,6);  print "";
for t in [1..#L] do
  rep:=L[t];  f:=CosetAction(G3,rep);  Q:=Image(f);
  if (Order(Q) eq 120) then K:=rep;end if;
  end for;

print AbelianQuotientInvariants(Core(G3,K));
R:=Rewrite(G3,K); // print R;

L:=LowIndexSubgroups(R,5);

graphs:=[];
for t in [1..#L] do
  rep:=L[t];  f:=CosetAction(G3,rep);  Q:=Image(f);
  if (Order(Q) in [100..15000]) then
    ff:=CosetAction(Q,sub<Q|f(h),f(p),f(q)>); QQ:=Image(ff);
    edges:={1,1^ff(f(a))}^QQ;  atg:=Graph<Degree(QQ)|edges>;  okm:=false;
    for gr in graphs do if IsIsomorphic(atg,gr) then okm:=true;break gr;end
    if;end for;
    if not(okm) then Append(~graphs,atg);
      print "Index",Index(R,L[t]),": Symmetric graph of order",Order(atg),

      "valency",Degree(Rep(VertexSet(atg))),"girth",Girth(atg),"diameter",D
      iameter(atg);
      end if;
    end if;
  end for;

**** Magma output ****

Magma V2.15-15    Fri Jul  1 2011 20:12:56 on mcon007-01 [Seed = 778952846]
Type ? for help.  Type <Ctrl>-D to quit.
>
> G3<h,p,q,a>:=Group<h,p,q,a | h^3,p^2,q^2,a^2,(p,q),(h,p),(q*h)
^2,a*p*a*q>;
>
> L:=LowIndexSubgroups(G3,6);  print "";

> for t in [1..#L] do
for>   rep:=L[t];  f:=CosetAction(G3,rep);  Q:=Image(f);
for>   if (Order(Q) eq 120) then K:=rep;end if;
for>   end for;
>
> print AbelianQuotientInvariants(Core(G3,K));
[ 0, 0, 0, 0, 0, 0 ]
> R:=Rewrite(G3,K); // print R;
>
> L:=LowIndexSubgroups(R,5);
>
> graphs:=[];
> for t in [1..#L] do
for>   rep:=L[t];  f:=CosetAction(G3,rep);  Q:=Image(f);
for>   if (Order(Q) in [100..15000]) then
for|if>     ff:=CosetAction(Q,sub<Q|f(h),f(p),f(q)>); QQ:=Image(ff);
for|if>     edges:={1,1^ff(f(a))}^QQ;  atg:=Graph<Degree(QQ)|edges>;  okm:=
fal\
se;
for|if>     for gr in graphs do if IsIsomorphic(atg,gr) then okm:=
true;break g\
r;end if;end for;
for|if>     if not(okm) then Append(~graphs,atg);
for|if|if>       print "Index",Index(R,L[t]),": Symmetric graph of
order",Orde\
```

```
r(atg),
for|if|if|print>
"valency",Degree(Rep(VertexSet(atg))),"girth",Girth(atg\
),"diameter",Diameter(atg);
for|if|if>       end if;
for|if>     end if;
for>   end for;
Index 1 : Symmetric graph of order 10 valency 3 girth 5 diameter 2
Index 2 : Symmetric graph of order 20 valency 3 girth 6 diameter 5
Index 4 : Symmetric graph of order 40 valency 3 girth 8 diameter 6
Index 5 : Symmetric graph of order 1250 valency 3 girth 16 diameter 10
>
> quit;

Total time: 0.700 seconds, Total memory usage: 25.61MB

// Note: The last of these examples is the largest known 3-valent graph of
diameter 10
```