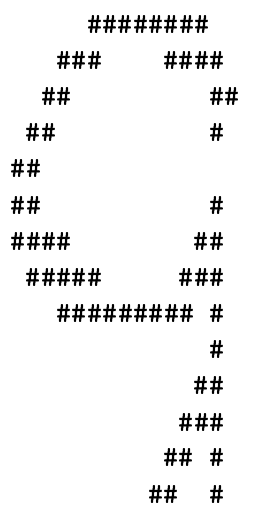
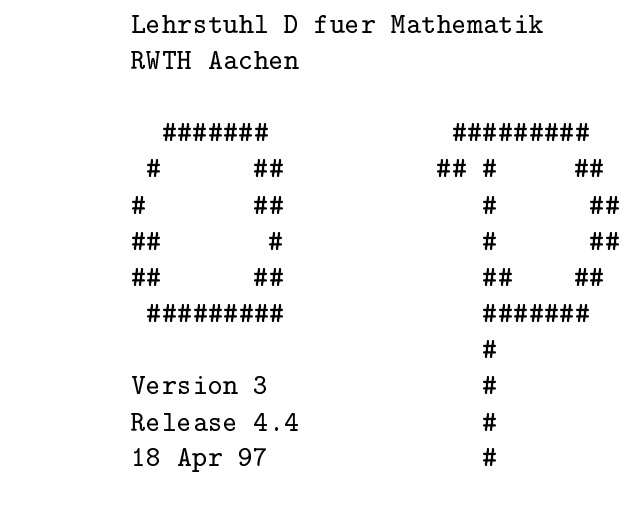


Lehrstuhl D fuer Mathematik
RWTH Aachen



Version 3
Release 4.4
18 Apr 97

Alice Niemeyer, Werner Nickel, Martin Schoenert
Johannes Meier, Alex Wegner, Thomas Bischops
Frank Celler, Juergen Mnich, Udo Polis
Thomas Breuer, Goetz Pfeiffer, Hans U. Besche
Volkmar Felsch, Heiko Theissen, Alexander Hulpke
Ansgar Kaup, Akos Seress, Erzsebet Horvath
Bettina Eick

For help enter: ?<return>

```
gap> 2+3/2-5*E(12)^5;
-7/2*E(12)^4-7/2*E(12)^8+5*E(12)^11
gap> E(12)^5;
-E(12)^11
gap> ZumbroichBase(12,1);
[ 4, 7, 8, 11 ]

gap> z := Z(9);
Z(3^2)
gap> z^4+z^0;
0*Z(3)
gap> z^4+z^3;
Z(3^2)^5
gap> z^2+z^3;
Z(3)

gap> a := (1,2,3,4,5,6,7,8,9,10,11);
( 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11)
gap> b := (3,7,11,8)(4,10,5,6);
( 3, 7,11, 8)( 4,10, 5, 6)
gap> a*b^2;
( 1, 2,11)( 3, 5,10)( 6, 8, 9)
gap> G := Group(a,b);
Group( ( 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11), ( 3, 7,11, 8)( 4,10, 5, 6) )
gap> Size(G);
7920
```

```
gap> Display( CharTable(G) );

      2  4   .   .  3  4  3  3  1  .  1
      3  2   .   .  .  1  .  .  2  .  1
      5  1   .   .  .  .  .  .  .  1  .
     11  1   1   1  .  .  .  .  .  .  .

         1a 11a 11b 4a 2a 8a 8b 3a 5a 6a
     2P  1a 11b 11a 2a 1a 4a 4a 3a 5a 3a
     3P  1a 11a 11b 4a 2a 8a 8b 1a 5a 2a
     5P  1a 11a 11b 4a 2a 8b 8a 3a 1a 6a
     7P  1a 11b 11a 4a 2a 8b 8a 3a 5a 6a
    11P  1a  1a  1a 4a 2a 8a 8b 3a 5a 6a

X.1      1   1   1  1  1  1  1  1  1  1
X.2     10  -1  -1  2  2  .  .  1  . -1
X.3     10  -1  -1  . -2  B -B  1  .  1
X.4     10  -1  -1  . -2 -B  B  1  .  1
X.5     11   .   . -1  3 -1 -1  2  1  .
X.6     16   A  /A  .  .  .  . -2  1  .
X.7     16  /A   A  .  .  .  . -2  1  .
X.8     44   .   .  .  4  .  . -1 -1  1
X.9     45   1   1  1 -3 -1 -1  .  .  .
X.10    55   .   . -1 -1  1  1  1  . -1

A = E(11)^2+E(11)^6+E(11)^7+E(11)^8+E(11)^10
  = (-1-ER(-11))/2 = -1-b11
B = E(8)+E(8)^3
  = ER(-2) = i2
```

```
gap> l := [-7,1,5];
[ -7, 1, 5 ]
gap> Length(l);
3
gap> l[3];
5
gap> l[4] := 9;
9
gap> l;
[ -7, 1, 5, 9 ]
gap> l[25] := -10;
-10
gap> l;
[ -7, 1, 5, 9,,,,,,,,,,,,,,,,,,,,, -10 ]
gap> l[5];
Error, List Element: <list>[5] must have a value
gap> l{[1,3]};
[ -7, 5 ]
gap> A := [[2,-3],[1,0],[-2,5]];
[ [ 2, -3 ], [ 1, 0 ], [ -2, 5 ] ]
gap> Length(A);
3
gap> Length(A[1]);
2
gap> A{[1,3]}{[2]};
[ [ -3 ], [ 5 ] ]
```

```
gap> myFactorial := function(n)
> local i,f;
> if n<0 then return false; elif n=0 then return 1;
> else f := 1;for i in [2..n] do f := f*i;od;fi;return f;end;
function ( n ) ... end

gap> Print(myFactorial);
function ( n )
    local  i, f;
    if n < 0  then
        return false;
    elif n = 0  then
        return 1;
    else
        f  :=  1;
        for i  in [ 2 .. n ]  do
            f  :=  f * i;
        od;
    fi;
    return f;
endgap> myFactorial(300);
306057512216440636035370461297268629388588804173576999416776741259476533176716\
867465515291422477573349939147888701726368864263907759003154226842927906974559\
841225476930271954604008012215776252176854255965356903506788725264321896264299\
365204576448830388909753943489625436053225980776521270822437639449120128678675\
368305712293681943649956460498166450227716500185176546469340112226034729724066\
333258583506870150169794168850353752137554910289126407157154830282284937952636\
580145235233156936482233436799254594095276820608062232812387383880817049600000\
00000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
gap> Print(Factorial);
function ( n )
    if n < 0  then
        Error( "<n> must be nonnegative" );
    fi;
    return Product( [ 1 .. n ] );
endgap>
gap> f := x->x^2;
function ( x ) ... end
gap> Print(f);
function ( x )
    return x ^ 2;
endgap>

gap> l1 := l{[1..4]};
[ -7, 1, 5, 9 ]

gap> List(l1, myFactorial);
[ false, 1, 120, 362880 ]

gap> Filtered(last,x->x<>false);
[ 1, 120, 362880 ]
```

```
gap> q := X(Rationals);q.name := "q";
X(Rationals)
"q"
gap> IsRec(q);
true
gap> RecFields(q);
[ "coefficients", "baseRing", "isPolynomial", "valuation", "domain",
  "operations", "name" ]
gap> q.coefficients;
[ 1 ]
gap> q.baseRing;
Rationals
gap> Size(Rationals);
"infinity"
gap> f := (q^3-7)*(6*q^5+q+2);
6*q^8 - 42*q^5 + q^4 + 2*q^3 - 7*q - 14
gap> f.coefficients;
[ -14, -7, 0, 2, 1, -42, 0, 0, 6 ]
gap> R := PolynomialRing(Rationals);
RationalsPolynomials
gap> q in R;
true
gap> f in R;
true
gap>?Records
...
```

```
gap> ?Size
    Size ____________________________________________________________ Domains

    'Size( <D> )'

    'Size' returns the  size of the domain <D>.  If <D> is  infinite, 'Size'
    returns the  string  '"infinity"'.   <D> may  also be a proper set  (see
    "Set"), in  which  case  the result  is  the length of this list.  'Size'
    will, however, signal an error if <D> is a list that is not a proper set,
    i.e., that is not sorted, or has holes, or contains duplicates.

        gap> Size( GaussianIntegers );
        "infinity"
        gap> Size( D12 );
        12

    The default function to compute the size of a domain is 'DomainOps.Size',
    which  computes the  set of  elements of  the  domain  with  the function
    'Elements' (see "Elements") and  returns the  length  of this  set.  This
    default function is overlaid in practically every domain.
```

```
gap> Print(Size);
function ( D )
    local  size;
    if IsSet( D )  then
        size  :=  Length( D );
    elif IsRec( D ) and IsBound( D.size )  then
        size  :=  D.size;
    elif IsDomain( D )  then
        D.size  :=  D.operations.Size( D );
        size  :=  D.size;
    else
        Error( "<D> must be a domain or a set" );
    fi;
    return size;
endgap>
gap> l := [1,2,-4];
[ 1, 2, -4 ]
gap> Size(l);
Error, <D> must be a domain or a set in
Size( l ) called from
main loop
brk>    # type <ctrl-D> to exit break loop
gap> IsSet(l);
false
gap> Set(l);
[ -4, 1, 2 ]
gap> Size(last);
3
```

```
gap> AllTwoGroups(Size,128);;
gap> Length(last);
2328
gap> AllTwoGroups(Size,8,IsAbelian,false);
[ Group( a1, a2, a3 ), Group( a1, a2, a3 ) ]
gap> Display(CharTable( last[2] ));
#W  Warning: Group has no name


     2  3  3  2  2  2

       1a 2a 4a 4b 4c
    2P 1a 1a 2a 2a 2a
    3P 1a 2a 4a 4b 4c

X.1     1  1  1  1  1
X.2     1  1 -1  1 -1
X.3     1  1  1 -1 -1
X.4     1  1 -1 -1  1
X.5     2 -2  .  .  .
```

```
?En
Help: EnSeveral sections match this topic:
Environment
Enlarging Lists
Endomorphisms for near-rings
Endomorphisms for groups
EndomorphismClasses
gap> ?Env
 Environment _____________________________________________ Environment
 This  chapter describes the  interactive  environment in  which  you  use GAP.

 The first sections describe  the main read  eval print loop and the break loop
 (see "Main Loop", "Break Loops", and "Error").

 The next section describes the commands you  can use to edit the current input
 line (see "Line Editing").

 The next sections describe the   GAP  help  system (see  "Help", "Reading
 Sections",  "Format of Sections",    "Browsing through   the   Sections",
 "Redisplaying a Section", "Abbreviating Section Names", "Help Index").

 The next sections  describe  the input and output functions  (see "Read",
 "ReadLib", "Print",  "PrintTo",  "AppendTo",  "LogTo", "LogInputTo", ... ).

 The  next  sections describe  the  functions that  allow you to   collect
 statistics about a computation (see "Runtime", "Profile").

 The last sections describe the functions that allow  you to execute other
 programs as subprocesses from within   GAP  (see "Exec" and "Edit").
```

```
gap> RequirePackage("chevie");
   WELCOME  to  the  CHEVIE  package,  Version 3  (Dec  1996)

      Meinolf Geck,  Frank Luebeck,   Gerhard Hiss,
      Gunter Malle,  Jean Michel, and Goetz Pfeiffer,
          Lehrstuhl D fuer Mathematik, RWTH Aachen,
          IWR der Universitaet Heidelberg,
          University of St. Andrews and
          Universite Paris VII

   This replaces the former weyl package. For first help type

          ?CHEVIE Version 3 -- a short introduction

gap> W := CoxeterGroup("A",2);
CoxeterGroup("A", 2)
gap> RecFields(W);
[ "cartan", "simpleRoots", "simpleCoroots", "semisimpleRank", "matgens",
  "roots", "N", "isDomain", "isGroup", "identity", "generators",
  "operations", "isPermGroup", "isFinite", "1", "2", "rootInclusion",
  "rootRestriction", "parentN", "isCoxeterGroup", "rank",
  "orbitRepresentative", "orbitRepresentativeElements", "rootLengths",
  "type", "coroots", "degrees", "size", "createArgs" ]
gap> W.cartan;
[ [ 2, -1 ], [ -1, 2 ] ]
gap> W.roots;
[ [ 1, 0 ], [ 0, 1 ], [ 1, 1 ], [ -1, 0 ], [ 0, -1 ], [ -1, -1 ] ]
gap> PermCoxeterWord(W,[1,2,1]);
(1,5)(2,4)(3,6)
```

```
gap> ?CoxeterLength

    CoxeterLength _____________________________ Elements in finite Coxeter groups

    'CoxeterLength( <W> , <w> )'

    returns the length of the permutation <w>, which corresponds to an
    element in the Coxeter group <W>, as a reduced expression in the standard
    generators.

        gap> W  :=  CoxeterGroup( "F", 4 );;
        gap> p  :=  PermCoxeterWord( W, [ 1, 2, 3, 4, 2 ] );
        ( 1,44,38,25,20,14)( 2, 5,40,47,48,35)( 3, 7,13,21,19,15)
        ( 4, 6,12,28,30,36)( 8,34,41,32,10,17)( 9,18)(11,26,29,16,23,24)
        (27,31,37,45,43,39)(33,42)
        gap> CoxeterLength( W, p );
        5
        gap> CoxeterWord( W, p );
        [ 1, 2, 3, 2, 4 ]

    This function requires the package  "chevie"  (see "RequirePackage").
```

```
gap> W := CoxeterGroup("H",3,"A",1);
CoxeterGroup("H", 3, "A", 1)
gap> PrintDynkinDiagram(W);
        5
H3    1 - 2 - 3
A1    4
gap> W.cartan;
[ [ 2, E(5)^2+E(5)^3, 0, 0 ], [ E(5)^2+E(5)^3, 2, -1, 0 ], [ 0, -1, 2, 0 ],
  [ 0, 0, 0, 2 ] ]
gap> PrintArray(last);
[ [             2,  E(5)^2+E(5)^3,              0,              0 ],
  [  E(5)^2+E(5)^3,              2,             -1,              0 ],
  [             0,             -1,              2,              0 ],
  [             0,              0,              0,              2 ] ]
gap> LongestCoxeterWord(W);
[ 1, 2, 1, 2, 1, 3, 2, 1, 2, 1, 3, 2, 1, 2, 3, 4 ]
gap> PermCoxeterWord(W,last);
( 1,17)( 2,18)( 3,19)( 4,20)( 5,21)( 6,22)( 7,23)( 8,24)( 9,25)(10,26)(11,27)
(12,28)(13,29)(14,30)(15,31)(16,32)
```

```
gap> CharTable(CoxeterGroup("A",3));
CharTable( "W( A3 )" )
gap> Display(last);
W( A3 )

      2    3    2    3    .  2
      3    1    .    .    1  .

        1111  211   22   31  4
     2P 1111 1111 1111   31 22
     3P 1111  211   22 1111  4

1111       1   -1    1    1 -1
211        3   -1   -1    .  1
22         2    .    2   -1  .
31         3    1   -1    . -1
4          1    1    1    1  1
```

```
gap> v := X(Rationals);;v.name := "v";;
gap> W := CoxeterGroup("H",3);;
gap> H := Hecke(W,v^2,v);
Hecke(CoxeterGroup("H", 3),[ v^2, v^2, v^2 ],[ v, v, v ])

gap> T := Basis(H,"T");
function ( arg ) ... end
gap> T([1])^2;
v^2T()+(v^2-1)T(1)

gap> T([1,2,3])^-1;
(-1+3v^-2-3v^-4+v^-6)T()+(v^-2-2v^-4+v^-6)T(1)+(v^-2-2v^-4+v^-6)T(2)+(v^-2-2v^\
-4+v^-6)T(3)+(-v^-4+v^-6)T(1,3)+(-v^-4+v^-6)T(2,1)+(-v^-4+v^-6)T(3,2)+v^-6T(3,\
2,1)

gap> C := Basis(H,"C");
function ( arg ) ... end
gap> C([1])*C([2,1]);
C(1)+C(1,2,1)

gap> T(C(last));
(-v^3-v)T()+(v+v^-1)T(1)+vT(2)-v^-1T(1,2)-v^-1T(2,1)+v^-3T(1,2,1)
gap> C(last);
C(1)+C(1,2,1)
gap> C(LongestCoxeterWord(W));
C(1,2,1,2,1,3,2,1,2,1,3,2,1,2,3)
gap> T(last);
-v^15T()+v^13T(1)+v^13T(2)+v^13T(3)-v^11T(1,2)-v^11T(1,3)-v^11T(2,1)- ...
```

```
 ?LeftCells
    LeftCells _________________________ Kazhdan-Lusztig polynomials and bases

    'LeftCells( <W> )'
    returns a list of pairs. The first component of each pair consists of the
    reduced  words in the Coxeter  group <W> which  lie in one left cell  C,
    the second component consists  of   the corresponding matrix of   highest
    coefficients   mu_(y,w),  where  y,w  are in  C.

gap> l := LeftCells(W);;
gap> Length(l);
22
gap> List(l,x->Length(x[1]));
[ 1, 6, 5, 8, 5, 6, 1, 5, 8, 5, 5, 6, 6, 5, 8, 5, 5, 8, 5, 6, 6, 5 ]
gap> l[3];
[ [ [ 1, 3, 2, 1 ], [ 2, 1, 3, 2, 1 ], [ 1, 2, 1, 3, 2, 1 ],
      [ 2, 1, 2, 1, 3, 2, 1 ], [ 3, 2, 1, 2, 1, 3, 2, 1 ] ],
  [ [ 0, 1, 0, 0, 0 ], [ 1, 0, 1, 0, 0 ], [ 0, 1, 0, 1, 0 ],
      [ 0, 0, 1, 0, 1 ], [ 0, 0, 0, 1, 0 ] ] ]
gap> LeftCellRepresentation(H,last);
[ [ [ -v^0, v, 0*v^0, 0*v^0, 0*v^0 ], [ 0*v^0, v^2, 0*v^0, 0*v^0, 0*v^0 ],
    [ 0*v^0, v, -v^0, v, 0*v^0 ], [ 0*v^0, 0*v^0, 0*v^0, v^2, 0*v^0 ],
    [ 0*v^0, 0*v^0, 0*v^0, 0*v^0, v^2 ] ],
  [ [ v^2, 0*v^0, 0*v^0, 0*v^0, 0*v^0 ], [ v, -v^0, v, 0*v^0, 0*v^0 ],
    [ 0*v^0, 0*v^0, v^2, 0*v^0, 0*v^0 ], [ 0*v^0, 0*v^0, v, -v^0, v ],
    [ 0*v^0, 0*v^0, 0*v^0, 0*v^0, v^2 ] ],
  [ [ -v^0, v, 0*v^0, 0*v^0, 0*v^0 ], [ 0*v^0, v^2, 0*v^0, 0*v^0, 0*v^0 ],
    [ 0*v^0, 0*v^0, v^2, 0*v^0, 0*v^0 ], [ 0*v^0, 0*v^0, 0*v^0, v^2, 0*v^0 ],
    [ 0*v^0, 0*v^0, 0*v^0, v, -v^0 ] ] ]
```

```
gap> W := CoxeterGroup("F",4);
CoxeterGroup("F", 4)
gap> PrintDynkinDiagram(W);
F4    1 - 2 > 3 - 4
gap> H := ReflectionSubgroup(W,[1,2,4]);
ReflectionSubgroup(CoxeterGroup("F", 4), [ 1, 2, 4 ])
gap> PrintDynkinDiagram(H);
A2    1 - 2
~A1    4

gap> ReducedRightCosetRepresentatives(W,H);
#nb. of cosets: 96
[ (), ( 2, 9)( 3,27)( 4, 7)( 5,11)(10,13)(12,15)(17,19)(20,22)(26,33)(28,31)
    (29,35)(34,37)(36,39)(41,43)(44,46), . . . ]
gap> List(last,x->CoxeterWord(W,x));
[ [  ], [ 3 ], [ 3, 4 ], [ 3, 2 ], [ 3, 2, 1 ], [ 3, 2, 4 ], [ 3, 2, 3 ],
  [ 3, 2, 1, 4 ], [ 3, 2, 1, 3 ], [ 3, 2, 4, 3 ], . . . ]
gap> Length(last);
96
```

```
gap> Display( InductionTable(H,W) );
Induction from A2x~A1 into F4
             | 111,11 111,2 21,11 21,2 3,11 3,2
-----------------------------------------------
phi_{1,0}    |      .     .     .    .    .   1
phi_{1,12}'' |      .     .     .    .    1   .
phi_{1,12}'  |      .     1     .    .    .   .
phi_{1,24}   |      1     .     .    .    .   .
phi_{2,4}''  |      .     .     .    .    1   1
phi_{2,16}'  |      1     1     .    .    .   .
phi_{2,4}'   |      .     .     .    1    .   .
phi_{2,16}'' |      .     .     1    .    .   .
phi_{4,8}    |      .     .     1    1    .   .
phi_{9,2}    |      .     .     1    2    1   2
phi_{9,6}''  |      .     .     2    1    2   1
phi_{9,6}'   |      1     2     1    2    .   .
phi_{9,10}   |      2     1     2    1    .   .
phi_{6,6}'   |      1     .     1    1    .   1
phi_{6,6}''  |      .     1     1    1    1   .
phi_{12,4}   |      1     1     2    2    1   1
phi_{4,1}    |      .     .     .    1    1   1
phi_{4,7}''  |      .     .     1    .    1   1
phi_{4,7}'   |      1     1     .    1    .   .
phi_{4,13}   |      1     1     1    .    .   .
phi_{8,3}''  |      .     .     1    1    2   2
phi_{8,9}'   |      2     2     1    1    .   .
phi_{8,3}'   |      .     1     1    2    .   1
phi_{8,9}''  |      1     .     2    1    1   .
phi_{16,5}   |      1     1     3    3    1   1
```

```
gap> ChevieCharInfo(W);
rec(
  charparams  :=  [ [ [ 1, 0 ] ], [ [ 1, 12, "''" ] ], [ [ 1, 12, "'" ] ],
      [ [ 1, 24 ] ], [ [ 2, 4, "''" ] ], [ [ 2, 16, "'" ] ],
      [ [ 2, 4, "'" ] ], [ [ 2, 16, "''" ] ], [ [ 4, 8 ] ], [ [ 9, 2 ] ],
      [ [ 9, 6, "''" ] ], [ [ 9, 6, "'" ] ], [ [ 9, 10 ] ], [ [ 6, 6, "'" ] ],
      [ [ 6, 6, "''" ] ], [ [ 12, 4 ] ], [ [ 4, 1 ] ], [ [ 4, 7, "''" ] ],
      [ [ 4, 7, "'" ] ], [ [ 4, 13 ] ], [ [ 8, 3, "''" ] ], [ [ 8, 9, "'" ] ],
      [ [ 8, 3, "'" ] ], [ [ 8, 9, "''" ] ], [ [ 16, 5 ] ] ],
  charnames  :=  [ "phi_{1,0}", "phi_{1,12}''", "phi_{1,12}'", "phi_{1,24}",
      "phi_{2,4}''", "phi_{2,16}'", "phi_{2,4}'", "phi_{2,16}''",
      "phi_{4,8}", "phi_{9,2}", "phi_{9,6}''", "phi_{9,6}'", "phi_{9,10}",
      "phi_{6,6}'", "phi_{6,6}''", "phi_{12,4}", "phi_{4,1}", "phi_{4,7}''",
      "phi_{4,7}'", "phi_{4,13}", "phi_{8,3}''", "phi_{8,9}'", "phi_{8,3}'",
      "phi_{8,9}''", "phi_{16,5}" ],
  a  :=  [ 0, 4, 4, 24, 1, 13, 1, 13, 4, 2, 4, 4, 10, 4, 4, 4, 1, 4, 4, 13, 3,
      9, 3, 9, 4 ],
  A  :=  [ 0, 20, 20, 24, 11, 23, 11, 23, 20, 14, 20, 20, 22, 20, 20, 20, 11,
      20, 20, 23, 15, 21, 15, 21, 20 ],
  b  :=  [ 0, 12, 12, 24, 4, 16, 4, 16, 8, 2, 6, 6, 10, 6, 6, 4, 1, 7, 7, 13,
      3, 9, 3, 9, 5 ],
  B  :=  [ 0, 12, 12, 24, 8, 20, 8, 20, 16, 14, 18, 18, 22, 18, 18, 20, 11, 17,
      17, 23, 15, 21, 15, 21, 19 ],
  positionId  :=  1, positionSgn  :=  4, positionRefl  :=  17,
  kondo  :=  [ "1_1", "1_2", "1_3", "1_4", "2_1", "2_2", "2_3", "2_4", "4_1",
      "9_1", "9_2", "9_3", "9_4", "6_1", "6_2", "12", "4_2", "4_3", "4_4",
      "4_5", "8_1", "8_2", "8_3", "8_4", "16" ] )
quit;
```