GAP = Groups, Algorithms and Programming

- System for computational discrete algebra, in particular group theory
- Available via ftp - for free
- There is a copyright
  - it is prohibited to use it commercially

```
        ########            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>
```

History of GAP

1986 project started in Aachen
Joachim Neubüser with 4 diploma students

1988 first release - version 2.4

1992 second release - version 3.1
(substantial improvements)

1993 releases of version 3.2 and later 3.3

1994 release of version 3.4 - current version
19 authors (programming and manual)

Retirement of J. Neubüser - July 1997

- GAP moved to St. Andrews

- Steve Linton new programming head

- GAP council takes up most of the duties of J. Neubüser

New: GAP council acts as editorial bord for GAP packages!

Future of GAP

- Currently we are working on GAP 4
- A beta release is available
- GAP 4 will include substantial improvements

Aims of GAP

- For students: to learn group theory

- For researchers in pure algebra: a tool to work,
  e.g. large examples
  e.g a large number of examples
  e.g. using catalogues

- For researchers in computational algebra:
  e.g. provide a basis for their implementations
  e.g. a lot of known algos are available

Invitation: Give us your programms!

The GAP language - Pascal like

- Objects (integers, finite field elements, cyclotomics, ...)
- Lists and Records
- Assignments
- If statement (if - then - elif - else - fi)
- For loop (for - in - do - od)
- While loop (while - do - od)
- Repeat loop (repeat - until)
- Functions

```
gap> ? Lists

Lists _______________________________________________________________ Lists

Lists  are the  most  important way  to collect objects  and  treat  them
together.  A *list* is a collection  of elements.  A list  also implies a
partial mapping from the integers  to  the elements.   I.e.,  there is  a
first element of a list, a second, a third, and so on.

List constants are written by writing down the  elements in order between
square brackets '[', ']', and separating them with commas ','.  An *empty
list*, i.e., a list with no elements, is written as '[]'.

    gap> [ 1, 2, 3 ];
    [ 1, 2, 3 ]    # a list with three elements
    gap> [ [], [ 1 ], [ 1, 2 ] ];
    [ [  ], [ 1 ], [ 1, 2 ] ]    # a list may contain other lists

Usually a list has no holes, i.e., contain an element  at every position.
However, it is  absolutely  legal  to  have  lists with holes.  They are
created by leaving the entry between the  commas empty.  Lists with holes
are sometimes  convenient when  the  list  represents  a mapping  from  a
finite, but  not consecutive, subset  of the positive  integers.  We  say
that a list that has no holes is *dense*.

    gap> l := [ , 4, 9,, 25,, 49,,,, 121 ];;
    gap> l[3];
    9
    gap> l[4];
    Error, List Element: <list>[4] must have a value

It is most common  that a list contains  only elements of one type.  This
is not a must  though.

               . . . . . . . . . . . . . . . . . .
```

Compute some Mersenne Primes:

```
gap> mersenne := [ ];
[  ]

gap> for i in [ 1..5 ] do
>        m := 2^i - 1;
>        if IsPrime( m ) then
>            Add( mersenne, m );
>        fi;
>    od;

gap> mersenne;
[ 3, 7, 31 ]
```

Compute the first 11 Mersenne Primes:

```
gap> indices := [ ];
[  ]
gap> mersenne := [ ];
[  ]
gap> i := 1;
1

gap> while Length( mersenne ) < 12 do
>        m := 2^i - 1;
>        if IsPrime( m ) then
>            Add( mersenne, m );
>            Add( indices, i );
>        fi;
>        i := i + 1;
>    od;

gap> indices;
[ 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127 ]

gap> mersenne;
[ 3, 7, 31, 127, 8191, 131071, 524287, 2147483647,
  2305843009213693951,
  618970019642690137449562111,
  162259276829213363391578010288127,
  170141183460469231731687303715884105727 ]
```

Look at them as binary numbers:

```
gap> coeff := List( [1..9], x -> 2 );
[ 2, 2, 2, 2, 2, 2, 2, 2, 2 ]

gap> for j in indices do
>         Print( CoefficientsInt( coeff, j ),"\n");
> od;
[ 0, 0, 0, 0, 0, 0, 0, 1, 0 ]
[ 0, 0, 0, 0, 0, 0, 0, 1, 1 ]
[ 0, 0, 0, 0, 0, 0, 1, 0, 1 ]
[ 0, 0, 0, 0, 0, 0, 1, 1, 1 ]
[ 0, 0, 0, 0, 0, 1, 1, 0, 1 ]
[ 0, 0, 0, 0, 1, 0, 0, 0, 1 ]
[ 0, 0, 0, 0, 1, 0, 0, 1, 1 ]
[ 0, 0, 0, 0, 1, 1, 1, 1, 1 ]
[ 0, 0, 0, 1, 1, 1, 1, 0, 1 ]
[ 0, 0, 1, 0, 1, 1, 0, 0, 1 ]
[ 0, 0, 1, 1, 0, 1, 0, 1, 1 ]
[ 0, 0, 1, 1, 1, 1, 1, 1, 1 ]
```

Now we combine two lists in a record:

```
gap> marin := rec( indices := indices,
                   primes := mersenne );
rec(
  indices := [ 2, 3, 5, 7, 13, 17, 19, 31, 61, 89,
      107, 127 ],
  primes := [ 3, 7, 31, 127, 8191, 131071, 524287,
      2147483647,
      2305843009213693951,
      618970019642690137449562111,
      162259276829213363391578010288127,
      170141183460469231731687303715884105727 ] )
```

Write a function that adds the next Mersenne Prime to a record like ‘marin’:

```
gap> AddNextMersennePrime := function( pr )
>         local j, f, m;
>         j := pr.indices[ Length( pr.indices ) ];
>         f := false;
>         while not f do
>             j := j + 1;
>             m := 2^j - 1;
>             if IsPrime( m ) then
>                 Add( pr.indices, j );
>                 Add( pr.primes, m );
>                 return;
>             fi;
>        od;
> end;
function ( pr ) ... end
```

Use the new function:

```
gap> AddNextMersennePrime( marin );
gap> marin;
rec(
  indices := [ 2, 3, 5, 7, 13, 17, 19, 31, 61, 89,
      107, 127, 521 ],
  primes := [ 3, 7, 31, 127, 8191, 131071, 524287,
      2147483647,
      2305843009213693951,
      618970019642690137449562111,
      162259276829213363391578010288127,
      170141183460469231731687303715884105727,
      686479766013060971498190079908139321726943\
530014330540939446345918554318339765605212\
255964066145455497729631139148085803712198\
799971664381257402829111505715 1 ] )
```

Write a function that returns the next Mersenne Pri-
me:

```
gap> NextMersennePrime := function( j )
>         local f, m;
>         f := false;
>         while not f do
>             j := j + 1;
>             m := 2^j - 1;
>             if IsPrime( m ) then
>                 return rec( indices := j, prime := m );
>             fi;
>         od;
>     end;
function ( j ) ... end

gap> NextMersennePrime(2);
rec(
  indices := 3,
  prime := 7 )
```

GAP 3.4.4 consists of 5 parts - it comes with full source code

- GAP - Kernel (written in C)
- GAP - Library (written in GAP - language)
- Catalogues of various kind
- GAP - manual (in latex, on-line and html version)
- GAP - Share Libraries

Other useful things around GAP:

- GAP - Forum (e-mail forum for comments and bugs)
- GAP - Trouble (e-mail address for technical problems)
- GAP - www page (everyting about GAP) http://www-gap.dcs.st-and.ac.uk

Some remarks on GAP 4

- Faster than GAP 3
- More intellegent than GAP 3
- Easier to extend by new objects than GAP 3
- More objects available than in GAP 3
- Nearly upwards compatible with GAP 3