Mathematics 666  

Homework (due Sep. 1)  

A. Hulpke

Note: In this course groups always act from the right, i.e. the image of a point $\omega$ under a group element $g$ is $\omega^g$ and the image under $gh$ is $(\omega^g)^h$. Consequentially we also consider vector spaces as row vector spaces and act on them by matrix multiplication from the right.

1) Let 

$$G = \begin{pmatrix} -16 & -3 & -30 & 36 & 0 \\ 0 & 0 & 2 & 0 & -1 \\ 2 & 2 & 3 & -5 & 0 \\ -5 & 0 & -8 & 11 & -1 \\ 3 & 5 & 2 & -8 & 1 \end{pmatrix}, \begin{pmatrix} -3 & 24 & -80 & 0 & 32 \\ -1 & 1 & -6 & 2 & 2 \\ -1 & -3 & 2 & 3 & -1 \\ -2 & 9 & -33 & 2 & 13 \\ -2 & -6 & 2 & 6 & -1 \end{pmatrix}$$

be a matrix group, denote the natural representation of $G$ by $v$.

a) Show that the linear subspace $U = \{(1,3,0,-3,0),(0,0,1,0,-1)\} \leq \mathbb{Q}^5$ is invariant under the action of $G$.

**Hint:** If we write a list of vectors in GAP (so they are rows of a matrix), we can test with SolutionMat(vectors,a) whether the row vector $a$ is in the span of the rows.

b) Extend the given basis of $U$ to a basis of $\mathbb{Q}^5$ (by adding three suitable further vectors).

c) Determine a new representation $\varphi$ of $G$ which is equivalent to the natural representation $v$ and in which every matrix image has the form

$$\begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix}$$

d) Determine a 2-dimensional and a 3-dimensional representation of $G$.

e) Is the 2-dimensional representation obtained in d) reducible? (**Hint:** Check for common eigenvectors.)

2) Let $G$ be a finite permutation group acting on the points $\{1, \ldots, n\}$.

Let $F$ be a field and $F^n$ the $n$-dimensional (row) vector space over $F$ and $\{v_1, \ldots, v_n\}$ the standard basis of $F^n$.

a) Show that the map $g \mapsto (v_1 \mapsto v_{g(1)})$ defines a matrix representation of $G$ over $F$ of degree $n$.

b) Show that the vector $v_1 + v_2 + \cdots + v_n$ spans a $G$-invariant subspace $S$ of $F$. Determine the action of $G$ on $S$.

c) Let $\phi: F^n \rightarrow F$ the linear map that maps $\sum_{i=1}^n a_i v_i$ to $\sum_{i=1}^n a_i$. Show that ker $\phi$ is a $G$-invariant subspace $T$ of $F^n$ and determine a basis for $T$.

d) Determine intersection and span of $S$ and $T$. (Note that the result depends on whether the characteristic of $F$ divides $n$)

e) For $F = \mathbb{Q}$ and $G = S_3 = \{(1,2,3), (1,2)\}$ determine matrix images of the generators of $G$ for the action of $G$ on the (in this case 2-dimensional) subspace $T$. 

3) In the group algebra $\mathbb{Q}S_3$, calculate the product

$$(2 \cdot () + 1 \cdot (1, 2, 3)) \cdot (-1 \cdot (1, 2) + 1 \cdot (1, 3))$$

4) Let $F$ be a field, $G$ be a finite group and $N \triangleleft G$. Let $I$ be the (2-sided) ideal of the group algebra $FG$ generated by $\{n - 1 \mid n \in N, n \neq 1\}$. Show (for example by defining a suitable algebra homomorphism) that the factor algebra $FG/I$ is isomorphic to the group algebra $F(G/N)$ of the factor group $G/N$.

5) Get accustomed with the system GAP.

**An introduction to GAP**

You start GAP by clicking the GAP 4.7 (or newer) icon (on windows) or typing `gap` under Unix. The program will start up and you will get a text prompt, looking like this:

```
gap>
```

You now can type in commands (followed by a semicolon) and GAP will return the result.

To leave GAP you can either call `quit;` or type `<ctl>-d`

The editing commands are similar as in the EMACS editor (depending on the setup, also cursor keys might work):

- `<ctl>-p` gets the previous line. (If the cursor is not at the start of the line, it gets the last line that starts like the current one.)
- `<ctl>-n` gets the next line.
- `<ctl>-f` goes one character to the right.
- `<ctl>-b` goes one character to the left.
- `<ctl>-a` goes to the start of the line.
- `<ctl>-e` goes to the end of the line.
- `<ctl>-d` deletes the character under the cursor (insert is automatic).
- `<ctl>-k` deletes the rest of the line.

If you want to obtain a copy of your work, you can use `LogTo("filename");` which will create a file with the input and output you got. (Note: The file will appear in the directory in which GAP was started. Under Windows, you might have to give a full path such as `LogTo("c:/mydisk/home");` — however note that paths in GAP are always given with forward slashes in Unix style, even under Windows.) `LogTo();` will switch logging off again.

**Hints:**

- GAP is picky about upper case/lower case. `LogTo` is not the same as `logto`. 
• All commands end in a semicolon;

• Type ? followed by a subject name (and no semicolon...) to get the online help.

• If you get syntax errors, use <ctl>-p to get the line back, and correct the errors.

• Incorrect input, interruption of a calculation (by pressing <ctl>-c – this can be useful if you tried out a command that takes too long) or (heaven beware – if you encounter any please let me know) bugs can cause GAP to enter a so-called break-loop, indicated by the prompt brk>. You can leave this break loop with quit; or <ctl>-d.

• If you create larger input, it can be helpful to put it in a text file and to read it from GAP. This can be done with the command Read("filename");.

• By terminating a command with a double semicolon ;; you can avoid GAP displaying the result. (Obviously, this is only useful if assigning it to a variable.)

• everything after a hash mark (#) is a comment.

We now do a few easy calculations. If you have not used GAP before, it might be useful to do these on the computer in parallel to reading.

GAP knows integers of arbitrary length and rational numbers:

gap> 2^200-1;
1606938044258990275541962092341162602522202993782792835301375

gap> 123456/7891011+1;
2671489/2630337

gap> 17 mod 3;
2

gap> (9 - 7) * 5 = 9 - 7 * 5;
false

gap> 5/3<2;
true

You can assign numbers (or more general: every GAP object) to variables, by using the assignment operator :=. Once a variable is assigned to, you can refer to it as if it was a number. The special variables last, last2, and last3 contain the results of the last three commands.

gap> a:=2^16-1; b:=a/(2^4+1);
65535
3855

gap> 5*b-3*a;
-177330

gap> last+5;
-177325

gap> last+2;
-177323
Objects separated by commas and enclosed in square brackets form a list. Collections of objects are represented by such lists. Lists are also used to represent sets.

```
gap> l:=[5,3,99,17,2]; # create a list
[ 5, 3, 99, 17, 2 ]
gap> l[4]; # access to list entry
17

gap> l[3]:=22; # assignment to list entry
22

gap> l; # access to list entry
[ 5, 3, 22, 17, 2 ]

gap> Length(l);
5

gap> 3 in l; # element test
true

gap> 4 in l;
false

gap> Position(l,2);
5

gap> Add(l,17); # extension of list at end

gap> l;
[ 5, 3, 22, 17, 2, 17 ]

gap> s:=Set(l); # new list, sorted, duplicate free
[ 2, 3, 5, 17, 22 ]

gap> AddSet(s,4); # insert in sorted position

gap> AddSet(s,5); # and avoid duplicates

gap> s;
[ 2, 3, 4, 5, 17, 22 ]
```

Results that consist of several numbers are represented as list.

```
DivisorsInt(96);
[ 1, 2, 3, 4, 6, 8, 12, 16, 24, 28, 48, 96 ]
Factors(2^126-1);
[ 3, 3, 3, 7, 7, 19, 43, 73, 127, 337, 5419, 92737, 649657, 77158673929 ]
```

There are powerful list functions that often can save programming loops: List, Filtered, ForAll, ForAny, First. The notation \( i \rightarrow \text{xyz} \) is a shorthand for a one parameter function.

```
List(l,IsPrime);
```
[true, true, false, true, true]
gap> List(1,i -> i^2);
[25, 9, 9801, 289, 4]
gap> Filtered(1,IsPrime);
[5, 3, 17, 2]
gap> ForAll(1,i -> i>10); false
gap> ForAny(1,i -> i>10); true
gap> First(1,i -> i>10); 99

A special case of lists are ranges, indicated by double dots. They can also be used to create arithmetic progressions:

gap> l:=[10..100];
[10 .. 100]
gap> Length(l);
91

Lists are also used to form vectors and matrices:
A vector is simply a list of numbers. A list of (row) vectors is a matrix. GAP knows matrix arithmetic.

gap> vec:=[1,2,3,4];
[1, 2, 3, 4]
gap> vec[3]+2;
5
gap> 3*vec+1;
[4, 7, 10, 13]
gap> mat:=[[1,2,3,4],[5,6,7,8],[9,10,11,12]];
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
gap> mat*vec;
[30, 70, 110]
gap> mat:=[[1,2,3],[5,6,7],[9,10,12]];;
gap> mat^5;
[[289876, 342744, 416603], [766848, 906704, 1102091],
[1309817, 1548698, 1882429]]
gap> DeterminantMat(mat);
-4

The command Display can be used to get a nicer output:

gap> 3*mat^2-mat;
[[113, 130, 156], [289, 342, 416], [492, 584, 711]]
gap> Display(last);
The expression $E(n)$ is used to denote the $n$-th root of unity ($e^{\frac{2\pi i}{n}}$):

\[
gap> \text{root5:=} E(5) - E(5)^2 - E(5)^3 + E(5)^4;;
\]
\[
gap> \text{root5^2;}
\]
5

To compute in finite fields, we have to create special objects to represent the residue classes (Internally, GAP uses Zech Logarithms and represents all nonzero elements as power of a generator of the cyclic group).

\[
gap> \text{gf:=GF(7);}\GF(7)
\]
\[
gap> \text{One(gf);}\quad Z(7)^0
\]
\[
gap> a:=6*\text{One(gf);}\quad Z(7)^3
\]
\[
gap> b:=3*\text{One(gf);}\quad Z(7)
\]
\[
gap> a+b;\quad Z(7)^2
\]
\[
gap> \text{Int(a+b);}\quad 2
\]

We can also form matrices over finite fields.

\[
gap> \text{mat:=}[[1,2,3],[5,6,7],[9,10,12]]*\text{One(im)};
\[
\begin{bmatrix}
Z(7)^0 & Z(7)^2 & Z(7) \\
Z(7)^5 & Z(7)^3 & 0*Z(7) \\
Z(7)^2 & Z(7) & Z(7)^5
\end{bmatrix}
\]
\[
gap> \text{mat}^-1+\text{mat};
\]
\[
\begin{bmatrix}
Z(7)^4 & Z(7)^4 & Z(7)^4 \\
Z(7)^3 & Z(7)^0 & Z(7)^5 \\
Z(7) & Z(7)^0 & Z(7)^3
\end{bmatrix}
\]

**Groups and Homomorphisms**

We can write permutations in cycle form and multiply (or invert them):

\[
gap> a:=(1,2,3,4)(6,5,7);
(1,2,3,4)(5,7,6)
\]
\[
gap> a^2;
(1,3)(2,4)(5,6,7)
\]
\[
gap> a^-1;
\]
\[(1,4,3,2)(5,6,7)\]
gap> b:=(1,3,5,7)(2,6,8);; a*b;

**Note:** GAP multiplies permutations *from left to right*, i.e. \((1,2,3)(2,3)=(1,3)\). (This might differ from what you used in prior courses.)

A group is generated by the command `Group`, applied to generators (permutations or matrices). It is possible to compute things such as `Elements`, `group Order` or `ConjugacyClasses`.

\[
gap> g:=Group((1,2,3,4,5),(2,5)(3,4));
\]
\[
\text{Group}([ (1,2,3,4,5), (2,5)(3,4) ])
\]
\[
\text{gap> Elements(g)};
\]
\[
[ (1), (2,5)(3,4), (1,2)(3,5), (1,2,3,4,5), (1,3)(4,5), (1,3,5,2,4),
  (1,4)(2,3), (1,4,2,5,3), (1,5,4,3,2), (1,5)(2,4) ]
\]
\[
\text{gap> Order(g)};
10
\]
\[
\text{gap> c:=ConjugacyClasses(g)};
[ (1)^G, (2,5)(3,4)^G, (1,2,3,4,5)^G, (1,3,5,2,4)^G ]
\]
\[
\text{gap> List(c,Size)};
[ 1, 5, 2, 2 ]
\]
\[
\text{gap> List(c,Representative)};
[ (), (2,5)(3,4), (1,2,3,4,5), (1,3,5,2,4) ]
\]

Homomorphisms can be created by giving group generators and their images under the map. (GAP will check that the map indeed defines a homomorphism first. One can use `GroupHomomorphismByImagesNC` to skip this test.)

\[
gap> g:=Group((1,2,3),(3,4,5));
\]
\[
\text{gap> mat1:=[ [ 0, -E(5)-E(5)^4, E(5)+E(5)^4 ], [ 0, -E(5)-E(5)^4, -1 ],
  [ 1, 1, E(5)+E(5)^4 ];
}\]
\[
\text{gap> mat2:=[ [ 1, 0, E(5)+E(5)^4 ], [ -E(5)-E(5)^4, 0, E(5)+E(5)^4 ],
  [ -E(5)-E(5)^4, -1, -1 ];
}\]
\[
\text{gap> img:=Group(mat1,mat2);};
\]
\[
\text{gap> hom:=GroupHomomorphismByImages(g,img,[(1,2,3),(3,4,5)],[mat1,mat2]);}
[ (1,2,3), (3,4,5) ] ->
\]
\[
\text{gap> Image(hom,(1,2,3,4,5))};
[ E(5)+E(5)^4,0,1 ], [ E(5)+E(5)^4,1,0 ], [ -1, 0, 0 ]
\]
\[
\text{gap> r:=[ [ 1, E(5)+E(5)^4, 0 ], [ 0, E(5)+E(5)^4, 1 ], [ 0, -1, 0 ];
}\]
\[
\text{gap> PreImagesRepresentative(hom,r)};
(1,4,2,3,5)
\]