BRIEF NOTES ON SINGULAR

If you want to download Singular for your PC or Mac go to:

http://www.singular.uni-kl.de and follow the instructions there.

If you are using linux and want to process a file named inputfile and save the Singular computation in the file outputfile, write the following at the prompt:

```
% Singular < inputfile > outputfile
```

Here starts a Singular session:

```
SINGULAR /
A Computer Algebra System for Polynomial Computations / version 2-0-3 <
by: G.-M. Greuel, G. Pfister, H. Schoenemann \ February 2002 FB
Mathematik der Universitaet, D-67653 Kaiserslautern \ >
```

The prompt to enter commands in Singular is >

The first step is to declare a ring

```
> ring R = 0, (x,y,z), dp;
```

You must specify that it is a ring, the name, and then three attributes: the characteristic (in this case zero), the names of the variables and an order. Here are how to specify some important orders:

Global orders:
dp = grevlex, Graded Reverse Lexicographic
lp = lex, Lexicographic
Dp = deglex, Degree Lexicographic

Local orders:
ds = negative Graded Reverse Lexicographic.
ls = Negative Lexicographic
Ds = Negative Degree Lexicographic

If you want to have variable names x(1),...,x(15) you don't need to type them all, you can simply specify: x(1..15)

```
> ring R1 = 0, (x(1..15)), lp;

> R1;
// characteristic : 0
// number of vars : 15
// block 1 : ordering lp
// : names x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8)
x(9) x(10) x(11) x(12) x(13) x(14) x(15)
// block 2 : ordering C
```

Note that you can check the "parameters" of your ring by typing its name followed by ; (every Singular command must end with ;)

You can also consider the ring of polynomials with coefficients in the field of rational functions on some parameters. For example, the ring

```
> ring R2 = (0,a,b,c), (x,y,z), dp;
> R2;
// characteristic : 0
// 3 parameter : a b c
// minpoly : 0
// number of vars : 3
// block 1 : ordering dp
// : names x y z
// block 2 : ordering C
```

is a polynomial ring in the variables (x, y, z) with coefficients which may be rational on the parameters (a,b,c).

The last ring you defined is the active ring. If you want to go back to a previous ring you can use the command setring

```
> setring R;
> R;
// characteristic : 0
// number of vars : 3
// block 1 : ordering dp
// : names x y z
// block 2 : ordering C
```

You can now define polynomials or ideals in the active ring:

```
> poly f = x^3*y^2 - 2*x*y;
> f;
x3y2-2xy
```

Note how Singular deals with powers, writing them to the right of the variable. You can also input in that way but it's not recommended!

If you use variables from another ring, Singular won't like it:

```
> poly g = x(1)^2;
? `x(1)` is not defined
? error occurred in STDIN line 15: `poly g = x(1)^2;`
? expected poly-expression. type 'help poly;'
```

You must change to ring R1 for this to make sense:

```
> setring R1;
> poly g = x(1)^2;
> g;
x(1)^2
```

Or, instead, if you want to work with the polynomial g, defined in the ring R, in the new ring R1, you can fetch it:

```
> poly g = fetch(R,g);
```

Even if the names of the variables are different, fetch will translate the first variable in R into the first variable in R1, etc.

Note that when the variable uses indices, Singular uses the ^ form for powers.

We define a ring in four variables and define an ideal:

```
> ring R = 0, (x,y,z,w), lp;
// ** redefining R **

> poly f1 = 3*x^2 + 2*y*z - 2*x*w;
> poly f2 = 2*x*z - 2*y*w;
> poly f3 = 2*x*y - 2*z - 2*z*w;
> poly f4 = x^2 + y^2 + z^2 - 1;
> ideal I = f1, f2, f3, f4;
> I;
I[1]=3x2-2xw+2yz
I[2]=2xz-2yw
I[3]=2xy-2zw-2z
I[4]=x2+y2+z2-1
```

Note that Singular describes an ideal by a list of its generators.

The command groebner computes a Groebner basis relative to a global order, in this case the lexicographic order:

```
> ideal J = groebner(I);
> J;
J[1]=96w6+212w5-116w4-493w3-225w2+36w
J[2]=24zw3+53zw2+25zw-4z
J[3]=1105z2+960w5+1928w4-2076w3-4338w2-189w
J[4]=3yw2+3yw-2z3-4zw2-4zw+2z
J[5]=2yzw+2yz-4z4-2z2w2+z2w+7z2
J[6]=27y2-16yz3+4yzw2-14yzw-2yz-12z2w2+24z2w+51z2+12w2-27
J[7]=3x+4y2w-6yzw-3yz+2z2w-2w
```

Note that this result means that w is the smallest variable. We can make x the smallest variable by redefining the ring:

```
> ring Rx = 0, (y,z,w,x), lp;
```

Since only the monomial order has changed, the ideal makes sense in this new ring as well so we can "bring it" from R to Rx:

```
> ideal Ix = imap(R,I);
> Ix;
Ix[1]=2yz-2wx+3x2
Ix[2]=-2yw+2zx
Ix[3]=2yx-2zw-2z
Ix[4]=y2+z2+x2-1

> ideal Jx = groebner(Ix);
> Jx;
Jx[1]=24x6+25x5-18x4-25x3-6x2
Jx[2]=10wx-144x5-102x4+134x3+87x2+10x
Jx[3]=4w2-6wx3-2wx2+4w+6x4-3x3-6x2
Jx[4]=24zx4+25zx3+6zx2
Jx[5]=4zw-120zx3-77zx2+4z
Jx[6]=z2-3wx2+w+3x3
Jx[7]=yx-zw-z
Jx[8]=yw-zx
Jx[9]=2yz-2wx+3x2
```

```
Jx[10]=y2+z2+x2-1
```

We can compute the dimension of the quotient R/J with the command vdim (note that we have to use a Groebner basis in the active ring)

```
> vdim(Jx);
12
```

We can also get a basis of the quotient with the command kbase:

```
> ideal K = kbase(Jx);
> K;
K[1]=x5
K[2]=x4
K[3]=zx3
K[4]=x3
K[5]=zx2
K[6]=x2
K[7]=zx
K[8]=x
K[9]=w
K[10]=z
K[11]=y
K[12]=1
```

Important: to **eliminate variables** is usually MORE COVENIENT to first compute a Groebner basis of a given ideal using the term order dp and then asking Singular to eliminate variables.

Consider again the previous example:

```
> ring R = 0, (x,y,z,w), dp;
> poly f1 = 3*x^2 + 2*y*z - 2*x*w;
> poly f2 = 2*x*z - 2*y*w;
> poly f3 = 2*x*y - 2*z - 2*z*w;
> poly f4 = x^2 + y^2 + z^2 - 1;
> ideal I = f1, f2, f3, f4;
> ideal Jnew = groebner(I);
> Jnew;
Jnew[1]=xz-yw
Jnew[2]=3y2-2yz+3z2+2xw-3
Jnew[3]=xy-zw-z
Jnew[4]=x2+y2+z2-1
Jnew[5]=12w3-6yz-11z2-6x-23w
Jnew[6]=5yw2-zw2+5yw+zw+2z
Jnew[7]=12xw2-9yz-4z2+18xw-12w2-3x-16w
Jnew[8]=12z2w+6xw2+3yz+5z2-3x-7w
Jnew[9]=2yzw-6z2w-2xw2-3z2+3w
Jnew[10]=2z3-3yw2+4zw2-3yw+4zw-2z
Jnew[11]=2yz2-2yw2+3zw2+3zw
> ideal K = eliminate(Jnew,x*y*z);
K;
K[1]=96w6+212w5-116w4-493w3-225w2+36w
```

Note that K[1] equals the polynomial J[1] in our previous computation with the lexicographic order lp, but this is usually much quicker.

Also note that the command **eliminate** has as inputs the **ideal** and then the **product of the variables we want to eliminate.**

Singular also allows us to compute the radical of an ideal but we must first load the library primdec.lib

```
> LIB "primdec.lib";
// ** loaded /usr/local/singular/2-0-3/LIB/primdec.lib
(1.98.2.10,2002/03/25)
// ** loaded /usr/local/singular/2-0-3/LIB/matrix.lib
(1.26.2.1,2002/02/20)
// ** loaded /usr/local/singular/2-0-3/LIB/ring.lib (1.17.2.1,2002/02/20)
// ** loaded /usr/local/singular/2-0-3/LIB/inout.lib
(1.21.2.3,2002/02/20)
// ** loaded /usr/local/singular/2-0-3/LIB/random.lib
(1.16.2.1,2002/02/20)
// ** loaded /usr/local/singular/2-0-3/LIB/poly.lib (1.33.2.5,2002/04/09)
// ** loaded /usr/local/singular/2-0-3/LIB/elim.lib (1.14.2.2,2002/02/20)
// ** loaded /usr/local/singular/2-0-3/LIB/general.lib
(1.38.2.7,2002/04/12)
> ideal Jred = radical(Jx);
> Jred;
Jred[1]=24x5+25x4-18x3-25x2-6x
Jred[2]=10wx-144x5-102x4+134x3+87x2+10x
Jred[3]=4w2-6wx3-2wx2+4w+6x4-3x3-6x2
Jred[4]=24zx3+25zx2+6zx
Jred[5]=4zw-120zx3-77zx2+4z
Jred[6]=z2-3wx2+w+3x3
Jred[7]=yx-zw-z
Jred[8]=yw-zx
Jred[9]=2yz-2wx+3x2
Jred[10]=y2+z2+x2-1

> vdim(groebner(Jred));
10

> kbase(std(Jred));
_[1]=x4
_[2]=x3
_[3]=zx2
_[4]=x2
_[5]=zx
_[6]=x
_[7]=w
_[8]=z
_[9]=y
_[10]=1
```

Note that Jred[1]=24x5+25x4-18x3-25x2-6x is the square-free part of Jx[1]=24x6+25x5-18x4-25x3-6x2.

You can find the normal form of an element f in the ring R relative to the Groebner basis Jx with the command reduce

```
> reduce(x^10 - y^2*z^3*w^7, Jx);
-1256585057693/440301256704zx3-78536582645/73383542784zx2
-15468025/7962624x5+576911/1327104x4+15468025/7962624x3+750193/1327104x2
```

Note that coefficients may get out of hand...

You could also use the command NF(f,Jx) (Normal form) to get the normal form of f relative to Jx.

One the main advantages of Singular is that it also allows you to compute with local orders, i.e. in local rings. If we use the order ds for example, we are computing in the localization of the polynomial ring at the origin and this allows us to compute multiplicities of zeroes at the origin:

```
> ring A = 0, (x,y), dp;
> ideal I = x^10 + x^9*y^2, y^8 - x^2*y^7;
> ideal J = groebner(I);
> vdim(J);
83
> vdim(groebner(radical(J)));
4
```

These computations tell us that the equations x^10 + x^9*y^2 = 0 and y^8 - x^2*y^7 = 0 have 83 solutions but only 4 distinct ones. Since clearly the origin is a common solution we may compute the multiplicity by passing to the localization:

```
> ring B = 0, (x,y), ds;
> ideal I0 = imap(A,I);
> ideal J0 = std(I0);
> vdim(J0);
80
```

This means that the origin has multiplicity 80 and the other three roots are simple (multiplicity one). Of course, in this example one could see by inspection that we had a standard basis for the negative revlex.

Note that for local orders the command for standard bases is std. You can also use std to obtain a Groebner bases when the order is global.

You can also define mixed orders, for instance:

```
> ring R = 0, (a,b,c,d,U,V,W), (Dp(4), Ds(3));
```

Then, Singular will use a globa term order in the first 4 variables and a local order in the last 3 variables. So, if I is an ideal here and J =std(I), reduce(g,J) = 0 iff there exists a nonzero polynomial h(U,V,W) such that hg belongs to I.

Other commands:

Give two ideals I, J:
quotient(I,J)= quotient ideal (I:J)
intersect(I,J) = the ideal I intersection J
saturate (I,J) = the ideal { f / f.g^m lies in I for all g in J and some m}
radical(I)

To solve zero dimensional systems over the complex numbers, here is a library called solve.lib (https://www.singular.uni-kl.de/Manual/4-0-2/sing_1681.htm#SEC1756).

Use the command

```
option(redSB);
```

to compute the reduced Groebner basis.