Computer Algebra, week 1, lecture 4:

Modular arithmetic.

iquo = integer quotient
irem = integer remainder

```
> iquo(80,12);
                                   6
> irem(80,12);
                                   8
```

Means: if you divide 80 by 12, it fits 6 times, and the remainder is 8. So 80 = 6*12 + 8.

```
> 80/12;
```

$$\frac{20}{3}$$

```
> evalf(%);
                              6.666666667
```

The command evalf computes a floating point approximation (evalf = EVALuate Floating) of an exact expression.
As you see above, there are different ways to divide integers. A division with / could produce an integer or a rational number. But there is also integer division, which always produces integers. In integer division, there is a quotient (iquo in Maple, short for: integer quotient. The command quo in Maple means quotient of polynomials and not integers) and there is a remainder irem (again rem in Maple is for polynomials, and irem is integer remainder, so irem is for integers).

In general, if you have positive integers a,b, then integer division a by b means finding two integers q and r such that a=q*b+r. Here q is the quotient and r is the remainder. Furthermore r must be smaller than b, and r must be greater or equal than 0. That makes the quotient and remainder uniquely defined.

```
> a:=80;
                                a := 80
> b:=12;
                                b := 12
> q:=iquo(a,b);
                                q := 6
> r:=irem(a,b);
                                r := 8
> a;
                                  80
> q*b+r;
                                  80
```

So indeed a = q*b+r.

```
> a:=2083475029337498275;
```

```
                         a := 2083475029337498275
> b:=1000;
                                b := 1000
> ‘Number of Digits of a‘ := length(a);
                       Number of Digits of a := 19
> q:=iquo(a,b);
                          q := 2083475029337498
> r:=irem(a,b);  # irem when b=10^3 gives you the last 3 digits
                                r := 275
> a = q*b + r;
               2083475029337498275 = 2083475029337498275
```

Now lets say we keep b fixed for a little while, and we take a couple of a’s.

```
> a1, a2, a3 :=2234523, 4574574567, 86547654765465;
             a1, a2, a3 := 2234523, 4574574567, 86547654765465
> r1, r2, r3 := seq(irem(a||i,b), i=1..3);
  # Note: This is for Maple 6 and 7
  # To make it work in Maple 5 you must replace the || by a dot .
  # So in Maple 5 you have a.i instead of a||i
‘|‘ unexpected
> a1*a2*a3;
                     884689442003174970864474956565
> irem(%,b);
                                  565
> r1*r2*r3;
                                r1 r2 r3
> irem(%,b);
                           irem(r1 r2 r3, 1000)
```

Hey, that’s the same.

"Computing modulo b" means: "whenever you see a number K >= b, replace it with irem(K,b)".

If we compute a1*a2*a3 and then reduce it modulo b (that means: take the remainder irem(a1*a2*a3,b)) we get the same result as when we first reduce a1,a2,a3 modulo b, then take the product, and then reduce again modulo b. Another example:

```
> b:=9;
                                 b := 9
> a1, a2, a3 :=1231, 3434, 123;
                     a1, a2, a3 := 1231, 3434, 123
> r1, r2, r3 := seq(irem(a||i,b),i=1..3);
‘|‘ unexpected
> a1+a2+a3;
```

$$4788$$

```
> irem(%,b);
```

$$0$$

```
> r1+r2+r3;
```

$$r1 + r2 + r3$$

```
> irem(%,b);
```

$$\text{irem}(r1 + r2 + r3, 9)$$

```
> 135*a1^2*a2 + 98*a3^12+a3;
```

$$1175134057174257519619332771$$

```
> irem(%,b);
```

$$6$$

```
> 135*r1^2*r2 + 98*r3^12+r3;
```

$$135\, r1^2\, r2 + 98\, r3^{12} + r3$$

```
> irem(%,b);
```

$$\text{irem}(135\, r1^2\, r2 + 98\, r3^{12} + r3, 9)$$

```
> irem(135,b)*r1^2*r2 + irem(98,b)*r3^12+r3;
```

$$8\, r3^{12} + r3$$

```
> irem(%,b);
```

$$\text{irem}(8\, r3^{12} + r3, 9)$$

Whenever you have an expression with products and sums, you can replace whatever you want in there by its remainder modulo b. The expression will then change, but its remainder modulo b will not. The following illustrates this, and illustrates the use of the Maple command **map**.

```
> V:=[112,2343,32445,1114,12345];
```

$$V := [112, 2343, 32445, 1114, 12345]$$

```
> map(hello,V);
```

$$[\text{hello}(112), \text{hello}(2343), \text{hello}(32445), \text{hello}(1114), \text{hello}(12345)]$$

```
> map(hello,V, x,y,z);
```

$[\text{hello}(112, x, y, z), \text{hello}(2343, x, y, z), \text{hello}(32445, x, y, z), \text{hello}(1114, x, y, z),$

$\quad \text{hello}(12345, x, y, z)]$

```
> map(x -> x^2, V);
```

$$[12544, 5489649, 1052678025, 1240996, 152399025]$$

```
> map(x -> 1/x, V);
```

$$\left[\frac{1}{112}, \frac{1}{2343}, \frac{1}{32445}, \frac{1}{1114}, \frac{1}{12345}\right]$$

```
> irem( V[1]*V[2]+V[3]*V[4]*V[5] , 123);
```

$$39$$

```
> V:=map(irem,V,123);
```

$$V := [112, 6, 96, 7, 45]$$

```
> irem( V[1]*V[2]+V[3]*V[4]*V[5] , 123);
```

$$39$$

The following is an example about a determinant, then irem. Then we first do irem's in the matrix, take the determinant, and do irem. And again the result is the same.

```
> A:=matrix(4,4,[seq(seq(i^j,i=11..14),j=4..7)]);
```

$$A := \begin{bmatrix} 14641 & 20736 & 28561 & 38416 \\ 161051 & 248832 & 371293 & 537824 \\ 1771561 & 2985984 & 4826809 & 7529536 \\ 19487171 & 35831808 & 62748517 & 105413504 \end{bmatrix}$$

```
> with(linalg):
Warning, new definition for norm
Warning, new definition for trace
> d := det(A);
```

$$d := 3997261151801229312$$

```
> b:=34;
```

$$b := 34$$

```
> irem(d,b);
```

$$20$$

```
> AA:=map( irem, A, b);
```

$$AA := \begin{bmatrix} 21 & 30 & 1 & 30 \\ 27 & 20 & 13 & 12 \\ 25 & 2 & 33 & 32 \\ 3 & 24 & 21 & 6 \end{bmatrix}$$

```
> dd := det(AA);
```

$$dd := 687024$$

```
> irem(dd,b);
```

$$20$$

Note: If we only wanted to know if matrix A is invertible or not, so if det(A) is zero or not, it would have been sufficient to compute irem(det(AA),b). That computation is faster because the entries of AA are smaller than those of A. It would have shown that the determinant of A is non-zero modulo b, and hence det(A) is non-zero, and A is invertible.

Instead of writing irem(dd,b) you can also write:

```
> dd mod b;
```

$$20$$

The command mod will apply irem on every integer in sight, except exponents of polynomials. The b we've used so far, we were computing modulo b, is called the modulus. It is often denoted by m instead of b.

```
> m:=9;
```

$$m := 9$$

```
> f:=293* x^101 + 2134 * x^12 -2 * x^3;
```

$$f := 293\,x^{101} + 2134\,x^{12} - 2\,x^{3}$$

```
> f mod m;
```

$$5\,x^{101}+x^{12}+7\,x^{3}$$

```
> v:=[123,4325345,15*x^15+12*x-1];
```

$$v := [123,\ 4325345,\ 15\,x^{15}+12\,x-1]$$

```
> v mod m;
```

$$[6,\ 8,\ 6\,x^{15}+3\,x+8]$$

Note: In Maple, if a<0 and b>0 then irem(a,b)<=0. In mathematics the usual definition of remainder is such that the remainder is always >=0. The command mod in Maple behaves like that definition, "a mod b" is the usual definition of the integer remainder, it’s >= 0.

```
> r:= -123 mod 9;
```

$$r := 3$$

```
> r:=irem(-123,9);
```

$$r := -6$$

```
> q:=iquo(-123,9);
```

$$q := -13$$

```
> q*9+r;
```

$$-123$$

Maple’s definition of iquo is such that the relation a*q+r=b still holds. You can also compute the quotient and remainder at the same time.

```
> restart;
> q:=iquo(12345,12,r);
```

$$q := 1028$$

```
> r;
```

$$9$$

```
> q:=iquo(12345,12,r);
Error, wrong number (or type) of parameters in function iquo
```

Do you remember from the worksheet on quotes what the reason is that it fails the second time?

The optional third argument of iquo and irem must be a name. Since r was evaluated to 9, the input really is 12345, 12, 9 and so the third argument is not a name. To fix this one should stop the evaluation of r, as follows:

```
> q:=iquo(12345,12,’r’);
```

$$q := 1028$$

```
> r;
```

$$9$$

Now lets take some polynomials, multiply, and reduce all entries modulo 5. Then afterwards, lets first reduce, then multiply, and then reduce again.

```
> p:=29;
```

$$p := 29$$

```
> f1 := x^4+randpoly(x,degree=3);
```

$$f1 := x^4 - 85\,x^3 - 55\,x^2 - 37\,x - 35$$

```
> f2 := x^4+randpoly(x,degree=3);
```

$$f2 := x^4 + 97\,x^3 + 50\,x^2 + 79\,x + 56$$

```
> f:=expand(f1*f2);
```

$$f := -4837\,x - 13033\,x^4 - 14350\,x^3 - 7753\,x^2 - 1960 + x^8 + 12\,x^7 - 8250\,x^6 - 9543\,x^5$$

```
> sort(f);
```

$$x^8 + 12\,x^7 - 8250\,x^6 - 9543\,x^5 - 13033\,x^4 - 14350\,x^3 - 7753\,x^2 - 4837\,x - 1960$$

```
> f mod p;
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

```
> F1:=f1 mod p;
```

$$F1 := x^4 + 2\,x^3 + 3\,x^2 + 21\,x + 23$$

```
> F2:=f2 mod p;
```

$$F2 := x^4 + 10\,x^3 + 21\,x^2 + 21\,x + 27$$

```
> F:=expand(F1*F2) mod p;
```

$$F := x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

The polynomial f is an element of Q[x]. It is reducible, it equals f1*f2. The polynomial F equals f mod p. It is "reducible modulo p" because there are polynomials F1, F2 such that F1*F2 is congruent to F modulo p (i.e. the have the same remainder modulo p). This is something that Maple uses to factor polynomial.

```
> f;
```

$$x^8 + 12\,x^7 - 8250\,x^6 - 9543\,x^5 - 13033\,x^4 - 14350\,x^3 - 7753\,x^2 - 4837\,x - 1960$$

```
> factor(f);
```

$$(x^4 - 85\,x^3 - 55\,x^2 - 37\,x - 35)\,(x^4 + 97\,x^3 + 50\,x^2 + 79\,x + 56)$$

```
> F;
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

```
> factor(F);
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

```
> % mod p;
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

```
> factor(F) mod p;
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

The command factor(F) mod p does two things: first it factors, then it reduces modulo p. The following command does something else:

```
> K:=Factor(F) mod p;
```

$$K := (x^4 + 2\,x^3 + 3\,x^2 + 21\,x + 23)\,(x^4 + 10\,x^3 + 21\,x^2 + 21\,x + 27)$$

The command factor(F) searches for polynomials f1,f2,.. such that their product is exactly equal to F.

However, the command Factor(f) mod p; searches for polynomials f1,f2,.. such that the product f1*f2*... is not necessarily equal to F, the product is only the same as F when you reduce F and the product modulo p.

```
> expand(K);
```

$$1050\,x + 365\,x^4 + 788\,x^3 + 1005\,x^2 + 621 + x^8 + 12\,x^7 + 44\,x^6 + 114\,x^5$$

Not the same as F.

```
> % mod p;
```

$$x^8 + 12\,x^7 + 15\,x^6 + 27\,x^5 + 17\,x^4 + 5\,x^3 + 19\,x^2 + 6\,x + 12$$

That’s equal to F.

If the modulus is m, then the command "mod m" reduces all integers to integers in the range 0..m-1.

```
> m:=40;
```

$$m := 40$$

```
> k:=1/7 mod 40;
```

$$k := 23$$

There is a unique integer k in the range 0..m-1 such that the product k*7 reduces to 1 modulo m. Therefore, 1/7 will be reduced to that integer k when computing modulo m.

```
> k*7 mod 40;
```

$$1$$

```
> 1/15 mod 40;
Error, the modular inverse does not exist
```

This leads to an error because there is no k such that k*15 can reduce to 1 modulo 40.

In general, there exists a number k such that k*a reduces to 1 modulo m if and only if igcd(a,m)=1.

```
> igcd(13,40);
```

$$1$$

```
> 1/13 mod 40;
```

$$37$$

```
> igcd(22,40);
```

$$2$$

```
> 1/22 mod 40;
Error, the modular inverse does not exist
```

A special case is when the modulus m is a prime number. In that case we usually use the letter p. Now whenever an integer is reduced modulo p, it ends up in the range 0..p-1. All integers in this range have igcd 1 with p, except the number 0.
So modulo m you can divide by numbers that have igcd 1 with m, but modulo a prime p you can divide by any non-zero element (note that by non-zero I mean: non-zero after it has been reduced modulo p. Every multiple of p reduces to 0 modulo p).

```
> p:=19;
```

$$p := 19$$

```
> v:=[seq(1/i,i=1..p-1)];
```

$$v := \left[1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{11}, \frac{1}{12}, \frac{1}{13}, \frac{1}{14}, \frac{1}{15}, \frac{1}{16}, \frac{1}{17}, \frac{1}{18}\right]$$

```
> w:=v mod p;
```

$$w := [1, 10, 13, 5, 4, 16, 11, 12, 17, 2, 7, 8, 3, 15, 14, 6, 9, 18]$$

```
> seq(w[i]*i, i=1..p-1);
```

$$1, 20, 39, 20, 20, 96, 77, 96, 153, 20, 77, 96, 39, 210, 210, 96, 153, 324$$

```
> [%] mod p;
```

$$[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

Another typical thing for prime numbers is that when a is non-zero modulo p, then a^(p-1) reduces to 1 modulo p, which is called Fermat’s little theorem (not the same as his famous "last theorem").

```
> {seq(i^(p-1) mod p,i=1..p-1)};  # Fermat’s little theorem holds:
```

$$\{1\}$$

```
> p:=51;
```

$$p := 51$$

```
> {seq(i^(p-1) mod p,i=1..p-1)}; # Fermat’s little theorem fails,
  which means that p=51 can not be a prime number.
```

$$\{13, 15, 16, 18, 19, 21, 25, 30, 33, 34, 36, 42, 43, 49, 1, 4, 9\}$$

```
>
```