# TMA4155 — Exercise 5

Kristian Gjøsteen

2005-10-18

**NB! This exercise will be in the computer lab.**

The numbers used in this exercise are available in the Maple worksheet `numbers07.mws`.

## Task 1 — Introduction to Maple

First of all, Maple has excellent on-line help. Try the command `?gcd` in Maple to see the help for the function `gcd`. What does it do?

Try the Maple commands `2+3;`, `2*3;`, `Pi;`, `evalf(Pi);`. Note that after every command, you need to type a semi-colon `;` for Maple to process the command.

We can assign values to variables using the assignment operator `:=`. Try the commands `a:=3;`, `b:=7;`, `a;`, `b;`, and `a*b;`.

Now enter the command `a:=7;`, then move the cursor up to the line with `a*b;` and press Enter on that line. What happens?

Load the Maple worksheet `numbers07.mws` (available from the course web page) and choose the menu entry Edit/Execute/Worksheet. (This is equivalent to pressing Enter on every line in the worksheet.)

Go back to your other command window, and type `t2_n1;`. All of the assignments made in the other worksheet are now available in the first worksheet.

The command **restart**; clears all assignments. Try **restart**; followed by `a*b;` to see the effect.

You can compute the remainder of $a$ when divided by $n$ as `a mod n;`. Try `74 mod 7;`, `2^33 - 10 mod 11;` and `2^82 + 5 mod 11`.

Note that Maple first evaluates the expression, then computes the remainder. This does not work for large exponents: Try `2^473298147978423 mod 11;`. The trick is to tell Maple that it needs to use modular exponentiation when computing the power, and you do that by using `&^`. Try `2&^473298147978423 mod 11;`.

A **for** loop in Maple looks like this:

```
for i from 1 to 100 do print(i); od;
```

We can compute the sum of the first 100 integers with the statements:

```
n := 0;
for i from 1 to 100 do n := n + i; od;
```

If you do not want to see the result of a computation, use a colon : instead of a semi-colon ;. (Try the effect with the above loops.)

Conditional statements have the form:

```
if a = 7 then a := 3; else a := 7; fi;
```

Inside a loop, we can have multiple statements:

```
for i from 1 to 100 do
  if gcd(i,29) <> 1 then print(i); fi;
  if gcd(i,31) <> 1 then print(i); fi;
od;
```

## Task 2 — Factoring

Factor the integers t2_n1, t2_n2, t2_n3 and t2_n4 using one or more of Pollard's $p-1$-method, Pollard's $\rho$-method or Fermat factorization.

## Task 3 — Primality testing

**a.** Apply the Fermat test to the numbers t3_n1, t3_n2, t3_n3 for a few different bases. Which of them may be prime?

**b.** Apply the Rabin-Miller test to the numbers t3_n1, t3_n2, t3_n3 for a few different bases. Which of them may be prime?

## Task 4 — RSA key generation

The Maple function rand(a..b)() returns random numbers between $a$ and $b$. (While this function is useless for serious cryptography, it is sufficient for this exercise.)

**a.** Do RSA key generation, with primes between $2^{511}$ and $2^{512}$, and the encryption exponent smaller than 18. Use the public key to encrypt a message, and the secret key to decrypt the ciphertext.

**b.** Encrypt the message 13 using the public key from the previous message. Without knowing the secret key, how would you recover the message from the ciphertext?