Before you start, in the *text mode*, enter

```
Your name
Date
```

Then, switch to *math mode*, enter

```
> with(student):with(plots):with(numtheory):
```

to load the student, plots and numtheory packages.

The while loop

In Lab 1, we tried to find the first integer with more than 6 divisors, and we used a `for` loop to achieve the task, when we `break` the loop once a number reaches more than 6 divisors. However, if say, we need to find the first number that has 50 factors, how should we specify the `for` loop? How far do we need to go?

Such a problem can be solved by using the `while` loop instead. Consider the same problem using the `for` loop:

```
> for i from 1 to 60 do
    if tau(i)>6 then print(i); break;
    else continue;
    fi;
  od;
```

And now, the while loop can be written as follows:

```
> i:=0;
  while(true) do
    i:=i+1;
    if(tau(i)>6) then break; fi;
  od:
  print(i);
```

In this while loop, we `break` when i has more than 6 divisors. Note that in every run of the loop (in between `while.. do .. od`) that the value of i is incremented by 1 (by the line `i:=i+1`). At the end of the loop, when a value is found, we print the value that breaks the loop by using the `print` command. Also note that the initial condition `i:=0;` is absolutely essential to tell *Maple* where the starting point for i should be.

Another way to write this while loop is as follows:

```
> i:=1;
  while(tau(i)<=6) do
    i:=i+1;
  od:
  print(i);
```

Note the difference here is that we put the condition to continue running after the keyword `while`. Hence, the value of i keeps on incrementing (inside the loop) until the condition `tau(i) <= 6` is not true anymore, that is, when `tau(i)> 6`. In other words, we are putting in the negation of the break condition in the `while(...)`

Exercises:

(1) Write the two while loops from above. Can you explain the difference between the initial condition `i:=0;` and `i:=1;` in the two different loops? (Note: they do not change the computations involved, but there is a subtle difference why the first loop starts at 0 and the second one starts at 1)

(2) Write a `while` loop to find the smallest power i such that 2^i is bigger than 1 million.

(3) From question 1, modify your code so that we have a **procedure**, that on input integer n, it outputs the smallest integer with exactly n divisors.

Planning a code

When we write a procedural code, the best approach is to first write a **pseudocode** detailing the steps to be taken, before proceeding to write the code. For example, suppose we need to write a procedure for the Euclidean Algorithm. Suppose the inputs are d_{-2}, d_{-1}, then, we need to do the following:

$$\begin{aligned}
d_{-2} &= a_0 d_{-1} + d_0 \\
d_{-1} &= a_1 d_0 + d_1 \\
&\vdots \\
d_{k-2} &= a_k d_{k-1} + d_k \\
d_{k-1} &= a_{k+1} d_k + 0
\end{aligned}$$

Then $\gcd(d_{-2}, d_{-1}) = d_k$.

Note that the intermediate steps are the same in every step, namely we need to evaluate, for two numbers m, n, that

$$m = qn + r$$

where r is the remainder, and q is the quotient, and that the values of m, n are the values of d's from the previous division.

Therefore, we can generate a while loop for this purpose:

```
while(remainder is not 0) do the following
  let m and n be the d's from the previous division
  find the quotient and remainder in m=qn+r
od
When remainder is 0, the gcd is the last non-zero remainder.
```

Now, we can incorporate the rest of the procedural code as follows:

```
findgcd:=proc(a,b)

  let m:=a, n:=b
  do the first division to find the remainder in m=qn+r
  while(remainder is not 0) do the following
    let m:=n and n:=r (so that we can do the next division)
    find the remainder in m=qn+r
  od
  RETURN(n) (which is the last non-zero remainder)

end:
```

Exercises:

(4) Use the above pseudocode to develop the procedure `findgcd`. You can use the function `mod` to find the remainder. Look at the help page for the use of `mod`.

(5) The *Maple* procedure `isprime(n)` returns `true` if n is prime, and `false` otherwise. Let a, b be positive integers that are co-prime. Write a procedure that finds the smallest integer n such that $a + bn$ is a prime.