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.

Observations of $\tau(n)$

In *Maple*, the function $\tau(n)$ (the number of divisors of n) can be calculated using the function `tau(n)`. In last lab, we used a for-loop to calculate a sequence of values of $\tau(n)$. If we want to record the sequence, we can also use the `seq` command:

```
> points:=[seq([n,tau(n)], n=2..50)];
```

Notice that in the output, we have a collection of ordered pairs $[n, \tau(n)]$, and we call this list `points`. With this list of points, we can then generate a **pointplot** of our results:

```
> pointplot(points);
```

(1) **(a)** Generate the list of values of $\tau(n)$ for $n = 2 \ldots 1000$. Suppress the output by using ":" instead of ";" Then, generate a plot of the values.

(b) What can you say about the distribution of values $\tau(n)$? What is the largest value of $\tau(n)$ for $n \leq 1000$?

(c) Part of this graph indicates the distribution of primes for $n = 2 \ldots 1000$. Can you identify which ones are primes?

The `proc()...end;` Command

Suppose we would like to evaluate the following function at various points:

$$f(x) = \begin{cases} x^2 - 4 & \text{if } x \geq 0 \\ x^3 - 3 & \text{if } x < 0 \end{cases}$$

One way is to define the two parts of $f(x)$ as functions $g(x)$ and $h(x)$ depending on whether $x \geq 0$ or $x < 0$. In other words, depending on the value of x, we will make the following decision:

If $x \geq 0$ then
  evaluate $x^2 - 4$
else (that is, when $x < 0$)
  evaluate $x^3 - 3$.

However, a simple decision loop will not mean anything unless the value of x is known. What if we need to evaluate f multiple times?

A programming feature, similar to the `for...do...od;` and the `if...then...fi;` command structures, is the *Maple* **procedure**. A procedure is a group of commands that performs a particular task. The `proc` command typically has input variables (in order to complete a task) and returns one or more answers (depending on the task). For example, the following is a procedue that computes the function f above:

```
> f:=proc(x)
    if x >= 0 then
      RETURN(x^2-4);
    else
      RETURN(x^3-3);
    fi;
  end;
```

Notice that the `proc ... end` command begins with stating the function name (`f`), and we *define* f as a *procedure*. Inside the procedure, depending on the input value of x, we `RETURN` the output value as calculated. To execute this procedure, say, we would like to evaluate $f(2)$, simply type

```
> f(2);
```

The same problem can also be written as follows:

```
> f:=proc(x)
    local output;
    if x >= 0 then
      output:=x^2-4;
    else
      output:=x^3-3;
    fi;
    RETURN(output);
  end;
```

In this procedure, we evaluate the correct output for the function f first, before returning the correct value at the end of the procedure. Within the procedure, you need to define any labels you use as `local`. Here, the variable *output* is a *local* variable inside the procedure only.

Exercises:

(3) Write a procedure f that evaluates the following function

$$f(x) = \begin{cases} x^3 - \sin(x) & \text{if } x \geq 0 \\ -2x^4 + 3x & \text{otherwise} \end{cases}$$

(4) Write a procedure called `ispositive` where the procedure returns "true" if the input value is positive, otherwise, it returns "false". Test the accuracy of your procedure by evaluating `ispositive(1), ispositive(-2), ispositive(0)`.

(5) Write a procedure called `largestd` that on input n (where $n \geq 2$), returns the value k where $2 \leq k \leq n$ with the largest value of $\tau(k)$. Test your code to verify the correctness of your code.