Curtin University
DEPARTMENT OF MATHEMATICS & STATISTICS

ACCELERATED MATHEMATICS UNITS MATH1017 AND MATH1021

Lab Sheet 1

Philosophy: There are algorithms in mathematics. So it makes sense for a mathematician to know something about computer programming. We are going to learn some generic features of computer languages, but we will use Maple in this unit.

Find **Maple 2016** on your computer and open it up.

Then click on the icon to **Open New Worksheet**.

Today, mainly you are expected to explore a little, and use **Maple**'s inbuilt documentation to learn how it does things. A number of you probably already have some familiarity with some computer language (**python** perhaps). It will probably be useful to work in groups where at least one student has such a familiarity.

One way of bringing up **Maple**'s documentation is to enter a *topic* behind a `?` after the `>` (which is **Maple**'s prompt).

Features of modern computer languages

Modern computer languages feature the following (among other things). Also, many computer languages have a statement *terminator* or *separator*. **Maple** requires statements to be *terminated* with a `;` or a `:` (read **Maple**'s documentation to find out why there are two possibilities, and how they differ).

1. *assignment* statements.
2. **if** statement.
3. **for** statement.
4. **while** statement.
5. **repeat** statement.
6. **break** statement.
7. *function*s.
8. *procedure*s.

Maple has a **for** statement with many optional bits which allows it to behave as each of 3, 4 (and with 6) as 5. And **proc** does both 7 and 8.

The idea is a student should get in the practice of reading the documentation to find out how to do each of 1 to 8.

For 1, try: `?assignment`

This will bring up a new window with a description of the syntax of an *assignment* statement

A lot of the programs, that you will be asked to write will already have an inbuilt **Maple** function that will do it, but thats not the point. The student needs to get the practice of writing it for themselves. Theres probably lots of ways to cheat, but then you wont learn anything.

Exercise. Write a function (use **proc**) to calculate factorials. (Of course, you could just type: 3!; and hit ⟨Enter⟩, since it's an inbuilt function of **Maple**, but, as mentioned above, that's not the point!)

This exercise can be done iteratively, which would use **for**, or it can be coded recursively (that maybe too much to ask), but that would use the recurrence property:

$$(n+1)! = (n+1)n!.$$

Saving code

One *can* save a worksheet, but I recommend you keep it simple. Once you have a fragment of code that you want to save, save it to a text file (you can use **Notepad** and, say, save it on your thumbdrive, which should probably come up as `E:`).

Try opening up **Notepad** and create a file with the text:

```
prod := 1;
for i to 5 do
  prod := prod * i;
od;
```

Save it to: `E:\loop.txt`
To read this into **Maple**, you need to use **read** which expects a filename in a pair of double-quotes, but for **Maple** the backslash has special meaning; so it needs to be "escaped" by another backslash, i.e. after the `>` prompt you need to enter:

```
read "E:\\loop.txt"
```

(and then follow that by pressing the ⟨Enter⟩ key).
Note: there is no semicolon (or colon) at the end, here!!!
Alternatively, you can pretend you are on a UNIX system and use slash (just one):

```
read "E:/loop.txt"
```