Worksheet Six

"Now that is what I call thinking", said Majikthise, "Why do we never think of things like that?", "Dunno", said Vroomfondel in an awed whisper, "I think our brains must be too highly trained".

Douglas Adams, The Hitch Hikers Guide to the Galaxy 1979

Unit Learning Outcomes Addressed by this worksheet: 1 & 2

For all of the exercises below, clearly state the assertions which will be valid for each control structure used. Make electronic copies of all of your algorithms. Place these, along with your Java code, in your P06 directory.

Exercise One.

Modify your pseudo code algorithm to the grade calculation problem so that your algorithm loops when inputting the final mark and the number of assessments until it has valid input. The simplest way to achieve this is to introduce two sub modules (one for input of the final mark and the other for input of the number of assessments. Note that you will also have to remove some of your IF-THEN statements.

Exercise Two.

Check that your algorithm will function correctly by using the test data that your specified in worksheet five.

Exercise Three.

Copy your Java implementation of the grade calculation algorithm from worksheet five and modify it so that it matches your algorithm from exercise two.

Exercise Four.

To calculate e^x Java provides the method Math.exp(double x). e^x can also be calculated using the series below:

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \frac{x^8}{8!} + \frac{x^9}{9!} + \ldots$$

Design a pseudo code algorithm for a sub module called eToTheX(). The sub module has two IMPORT parameters. The first is an integer value which specifies the number of terms and the second is a real value which contains the value of x. When calculating factorials, use the example in the lecture notes to guide you. There is a problem if you use integers to do the factorial calculation. What is the problem? What is the simplest fix for the problem?

Exercise Five.

Design a main algorithm which will:

- Input a value for x.
- Repeatedly call your eToTheX sub module with the number of terms varying from 5 to 100.
- Output the result of the eToTheX call and next to that output the value of Math.exp()

Exercise Six.

Translate your pseudo code main and eToTheX sub module into Java.

Exercise Seven.

Modify your algorithm by designing a new eToTheX sub module which IMPORTs a tolerance instead of the number of terms. The new algorithm stops calculating new terms when the last term calculated is less than the tolerance value.

Exercise Eight.

Modify your main sub module so that after testing the first eToTheX submodule it performs similar tests on the new one. The tolerance should vary from 0.1 to 0.000001.

Exercise Nine.

Finally translate the resulting pseudo code into Java.