Curtin University — Discipline of Computing

Programming Design and Implementation (PDI (COMP1007))

2020, Semester 1

Test

Weight: 30% of the unit mark.

Released: 12pm Wednesday, 29th April 2020
Last Upload: 12pm Thursday, 30th April 2020
Due Date: 4 hours after dowloaded from Blackboard

Note: The reason for the 24 hour window is to accomodate for uploading issues and any timetabling clashes. It is recommended that you download and work on the test as soon as it is made available. Your time (**4 Hours** *(240 minutes)*) starts as soon as you have downloaded the test from Blackboard.

Please read this front cover carefully. If any one of these are done incorrectly, it can result in heavy penalties.

This is a **OPEN BOOK test**. There are **FIVE (5) questions**. Answer all of them. You have **4 Hours** *(240 minutes)* from the time the test is downloaded from Blackboard to complete the test. There are **100 marks** available.

Write in Vim (Terminal Text Editor) on **either** the 314 Linux Labs (remotely) or the VDI solution (`https://mydesktop.curtin.edu.au`), or your own version of Linux. You will be required to submit your Bash History (in the same way that you did in each workshop).

Each Question is to be stored in its own **`.txt`** file named: **`Question*.txt`** where the **`*`** is replaced with the word version of the number (e.g., **`One`**, **`Two`**)

All your code must conform to practices emphasised in the lectures and practicals. All code must conform to Computing's coding standard.

You must work alone on this assessment. You must not communicate with anyone other than your lecturer or the Unit Coordinator regarding any aspect of this Assessment.

Note that "Zero Marks" rules apply.

All submissions will be subjected to rigorous testing for plagiarism, collusion and any other forms of cheating. You must cite any and all design/java from any source, including your own work submitted for a different assessment/workshop.

Upon Completion, you must **`tar`** and **`gzip`** your Declaration of Originality, **`BashHistory.txt`**, Design files (**`.txt`**) and Java file (**`.java`**), and submit to Blackboard (in the same way that you did in each workshop).

```
[user@pc]$ tar -cvzf <StudentID>_TestAnswers.tar.gz TestAnswersDirectory
```

Question 1 (10 marks)

Assume that all of the variables referred to are declared and initialised as below:

```
String stellar = "Southern Cross' formation";
double galaxy = 32.86, nova = 15.158;
int nebula = 372, redGiant = 46;
```

Evaluate each of the following Java expressions. You must assume that each expression is independent of the others.

Show each step of the working on a seperate line. Marks will not be awarded if working is not shown.

Do **not** use short circuit evaluation.

(a) `nebula / 2 + (int)galaxy` **[2 marks]**

(b) `redGiant + stellar + nebula` **[2 marks]**

(c) `2.0 * (double)((redGiant - 6) / 10)` **[2 marks]**

(d) `nebula % 12 - (int)(nova + (double)redGiant) * 2` **[2 marks]**

(e) `((nebula / 20 > 18) || (nova > 6.23)) && (redGiant % 2 == 0)` **[2 marks]**

Question 2 (10 marks)

Given the Java variable declarations below:

```
public static final double TOL = 0.01;
String carnivore;
double critter, organism, mammal;
int amphibian, predator;
```

Write Java boolean expressions for the following:

(a) Deciding if `carnivore` is the same as `"Hyena"`. **[2 marks]**

(b) Deciding if `predator` is greater than `40` or `amphibian` is between `5` and `19` exclusive. **[2 marks]**

(c) Deciding if `mammal` is between `122.0` and `649.0` inclusive. **[2 marks]**

(d) Deciding if `amphibian` is a multiple of `37`. **[2 marks]**

(e) Deciding if `organism` and `critter` are the same. **[2 marks]**

Question 3 appears on the next page

Question 3 (15 marks)

The algorithm below contains at least 3 errors, inefficiencies or redundancies. Describe three of them. You must state WHY each is a problem.
Note 1: The purpose of the algorithm is not relevant.
Note 2: There are more than 3 issues, you only need to provide 3.

```
count = 0
WHILE x < 1 AND x > 10 DO
    OUTPUT "Input must be between 1 and 10"
    INPUT x
    sum = sum + x
ENDWHILE
mean = sum / count
OUTPUT "Average is ", mean
```

Question 4 appears on the next page

Question 4 (45 marks)

Note: Some Mathematical constructs may/may not represent true information. Your task is to take what is stated below and convert that. Writing down any assumptions you may need to make.

Markup's constant (ξ) can be approximated with the sum of the product of the series below.

$$\xi = \sum_{i=1}^{n} \prod_{j=0}^{k} \left(\frac{i^2 * (j+1)}{4nk}\right)$$

Your task is to design an algorithm to approximate Markup's constant to an inputted number of terms. Your algorithm should perform the following steps:

- Input a number of terms to approximate ξ (`n`). Your algorithm should repeat the input until the number of values input is between 2 and 35 (including 2 and 35).
- Input a number `k` which is the precision for each term, must be between 5 and 100 (including 5 and 100).
- Calculate the Real value of each term of `n`, storing it in an array.
- After **all** of the individual terms have been calculated, calculate the final value of ξ, storing it in the last element of the array.
- Upon completion of all the calculations, the algorithm should output each value in the array to the user.

Remember: You do not need to understand the Maths - just implement the right hand side of the equation.

Note 1: You may assume that there is a function called `pow(Real, Real)` where the first argument is the `base` and the second is the `exponent`, that exists in your program.

Note 2: You **must** make good use of submodules, as emphasised in the lectures and worksheet exercises. You are **not** required to handle exceptions in this algorithm.

Question 5 (20 marks)

Convert the algorithm that you write for question 4 into a complete Java **program**.

Note 1: Marks will be reduced for the Java not matching the pseudo-code, poor formatting, not following the Curtin Coding Standard and incorrect Java syntax. Partial marks will be awarded for an incomplete translation.

Note 2: This Math API may be useful to you:

- `double r = Math.pow(double base, double exponent);`
  This method returns $base^{exponent}$

Note: Please store this Question in a file named `Markup.java`. (This supersedes the rules on the front cover)

End of Test