Curtin University — Discipline of Computing

Programming Design and Implementation (PDI (COMP1007))

2020, Semester 1

Mock Test B

Weight: 30% of the unit mark.

Note: The Discipline of Computing does **not** give out answers/marking guides for tests or exams.

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 mark 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 captain = "Janeway";
double warp = 9.975, weapons = 44.56;
int crew = 144, decks = 15;
```

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) `(int)warp + crew / 10` **[2 marks]**

(b) `crew + captain` **[2 marks]**

(c) `(double)((decks - 4) / 2)` **[2 marks]**

(d) `(int)(weapons + (double) decks) * 2 - crew % 5` **[2 marks]**

(e) `((crew / 10 < 10) || (warp < 9.99)) && (decks % 5 == 0)` **[2 marks]**

Question 2 (10 marks)

Given the Java variable declarations below:

```
public static final double TOL = 0.0001;
String lorca;
double tilly, burnham, saru;
int hugh, paul;
```

Write Java boolean expressions for the following:

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

(b) Deciding if `tilly` is a multiple of `8`. **[2 marks]**

(c) Deciding if `burnham` is between `10` and `23` inclusive. **[2 marks]**

(d) Deciding if `hugh` is less than `21` or `paul` is between `8` and `13` exclusive. **[2 marks]**

(e) Deciding if `tilly` and `burnham` 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.

```
1  found = true
2  elements = 0
3  INPUT elements
4
5  CREATE array OF INT, SIZE OF elements
6  INPUT searchNum
7
8  FOR ii = 0 TO elements INC BY 1
9      index = 0
10     IF array[ii] EQUALS elements
11         index = ii
12         found = true
13         BREAK
14     END IF
15 END FOR
16 OUTPUT "Integer found at" + index
```

Question 4 appears on the next page

Question 4 (45 marks)

Write a pseudo code algorithm that approximate the number pi (π) to a certain level of precision. This can be calculated as follows:

$$\frac{\pi}{4} = \sum_{k=0}^{max} \frac{\sin(\theta * (2k+1))}{2k+1}$$

and can be calculated as a series of:

$$\frac{\sin(\theta * (2(0)+1))}{2(0)+1} + \frac{\sin(\theta * (2(1)+1))}{2(1)+1} + \frac{\sin(\theta * (2(2)+1))}{2(2)+1} + ...$$

Where max is the number of iterations to calculate.

Your algorithm should perform the following steps:

- Input a number of terms to approximate π. Your algorithm should repeat the input until the number of values input is between 20 and 100 (including 20 and 100).
- Input a number θ between 0 and 360. This value will then need to be converted to radians, you may assume that a function called **`radians()`** exists which takes in a real value (θ) and converts it to its radian equivalent.
- Calculate the value of each term, storing it in an array.
- After **all** of the individual terms have been calculated, calculate the final value of π (not $\pi/4$), 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. In the denominator, the formula 2n+1 may be useful.

Note 1: You may assume that there is a function called **`sin(Real)`** and **`toRadians(Real)`** that exists within your program.

Note 2: You **must** make good use of submodules, as emphasised in the lectures and worksheet exercises.

Note 3: You are **not** required to handle exceptions in this algorithm.

Question 5 appears on the next page

Question 5 (20 marks)

Convert the algorithm that you write for question 9 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.toRadians(double degrees);`
  Converts an angle measured in degrees to an approximately equivalent angle measured in radians.
- `double s = Math.sin(double radians);`
  Returns the trigonometric sine of an angle in radians.

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

End of Test