Curtin University — Discipline of Computing

Programming Design and Implementation (PDI (COMP1007))

2020, Semester 1

Final Assessment

Weight: 40% of the unit mark.
Released: 8:30am Monday, 15th June 2020
Last Upload: 8:30am Tuesday, 16th June 2020
Due Date: 4 hours after dowloaded from Blackboard, within the 24hour window.

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

Please **don't** start your assessment at 8am on Tuesday 16th thinking you have 4 hours, you will **only** have 30 minutes.

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 **FOUR (4) questions**. Answer all of them. You have **4 Hours** *(240 minutes)* from the time the test is downloaded from Blackboard to complete the test (within the 24-hour window). 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.

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/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 code from any source, including your own work submitted for a different assessment.

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

Question 1 (35 marks)

Your task is to design a Number converter in Pseudocode. The functionality should be as follows:

- Prompt the user with the options to select either `Binary`, `Decimal`, `Octal` or `Hexadecimal`.
- Prompt the user to enter in a "numeric-string". *(Alphanumeric in the case of Hex)*
- If the user selects either Binary or Decimal, you are required to check if the "numeric-string" is a `Real` number.
  - If it is `Real`, you are just required to convert from `Decimal` to `Binary` or convert from `Binary` to `Decimal`
- For all other cases, you will need to then prompt the user what type they would like to output to either `Binary`, `Decimal`, `Octal` or `Hexadecimal`.
- Convert the "numeric-string" to the type that the user selected, then output it.

Use your discretion as to how you write your menu, we want to see your thought process.
You must use the methods shown in the Lecture slides. You may **not** use any in-built functionality that does this for you.

> **Note:** You should work with **`String`**'s for the most part, then when you want to do the conversion, parse it (see the example code below). Don't do direct conversion for `octal/hexadecimal`, do the `Binary` conversion first. Remember Modularity!

Question 2 (25 marks)

Convert the algorithm that you write for Question 1 into a complete Java **program**.

Note 1: If you have not attempted Question 1, you will not receive any marks for this.
Note 2: Remember the common mistakes from the Test.
Note 3: 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 4: The following snippets of java code may be useful to you:

```java
// Splitting a number in groups of 5 (from the right)
int grouping = 5;
String number = "01234567859";
for(int i = number.length(); i <= 0; i -= grouping)
{
    System.out.println(number.substring(PDIMath.max(i - grouping, 0), i);
}
// Looping through a String
for(int i = 0; i < number.length(); i++)
{
    System.out.println(number.charAt(i));
}
// Splitting on a "."
String someString = "100101010.10001101";
String[] splitString = someString.split("\\.");
```

Question 3 (15 marks)

You have been provided an algorithm in the Lecture slides for how to implement Levenshtein's Distance for String matching (Lecture 10). Your task is to design a complete program in Pseudocode for this purpose, using that algorithm.

Your program is required to take in two **`String`**'s from the user and output the "distance" between the two.

The submodule **`costOfSubstitution`** must check to see if the two **`Character`**'s are the same, if so it will return 0, and 1 if they are different.

You may need to modify your **`min`** algorithm in **`PDIMath`** to import 3 numbers instead of 2 and return the minimum of the 3.

Hint: Use your methods that we have designed throughout the semester to save you time. *(Just remember you must reference them)*

Note: You are **required** to re-download the Lecture notes, as they have been slightly modified for this Question.

Question 4 (25 marks)

Convert the algorithm that you write for question 3 into a complete Java **program**.

Note 1: If you have not attempted Question 3, you will not receive any marks for this.
Note 2: Remember the common mistakes from the Test.
Note 3: 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: Please store Question 2 in a file named **`NumberConverter.java`**.
Please store Question 4 in a file named **`Levenshtein.java`**.
(These supersede the rules on the front cover)

End of Test