Software Development 1, Coursework 1

This is <u>individual</u> assessed coursework. You are allowed to discuss this assessment with other students, but you should not copy their code, and you should not share your own code with other students. We will carry out plagiarism checks on submissions.

Objectives

This coursework contributes 10% of your overall mark for the course. It assesses your understanding of topics covered in Weeks 1, 2 and 3, including:

- Using GitLab to pull and push code
- Creating, storing to, and accessing variables
- Understanding the role of types
- Creating expressions from operators, variables and values
- Using conditional execution to implement decision making

These topics are all covered in the learning materials in Weeks 1, 2 and 3 on Canvas. It is also recommended that you work through the tutorial exercises from these weeks before attempting the coursework, since these help build the understanding you need to complete the coursework.

Submission

The **deadline** for completing this coursework is **<u>Monday 10th October (Week 5)</u>**. Make sure you do these two things:

1) **Commit your work to GitLab** before this deadline. If you don't do this, a late penalty may be applied. There are instructions on using GitLab in the Week 1 familiarisation tutorial.
2) **For students in Edinburgh, attend a lab session and ask someone to mark your work**. After they've done this, they will give you feedback on your work. You may get your work marked before the deadline or in the week following the deadline without a penalty, so long as you committed it to GitLab before the deadline.

<u>Do This First</u>: Fork and Import the GitLab Project

You will be adding your code to an existing project in GitLab. The first thing you need to do is fork this project, so that you'll be working on your own copy, and import it into Eclipse:

1. Visit https://gitlab-student.macs.hw.ac.uk/F27SA_2022-23/f27sa_2022-23_coursework-1
2. Fork the project
3. Import your forked project into Eclipse

! ***Don't forget to fork the project before you import it***
if you don't do this, you won't be able to save your changes back to Gitlab.

If you don't know how to do any of this, then make sure you first work through the Week 1 tutorial, which will take you through the process step by step. If you're having trouble getting it to work, then come to a timetabled lab session and talk to a lab helper, well before the submission deadline.

(continues on next page…)

The Behaviour of Your Programme

The GitLab project contains a single class, **DegreeAdvice.java**, where you are going to add your code. The programme will help students to understand how they are doing in their degree programme. When executed, it will ask the student which degree programme they are studying on and which marks they have for their courses in the current semester, and it will tell them their average mark and whether they have achieved the marks required to progress.

To keep things simple, you only need to consider that:

- Students are in the first semester of their first year
- All students take the four courses F27SA, F27ID, F27PX and F17LP
- Students are on either the BSc Computer Science or BSc Computer Systems programme.

To determine whether a student has achieved the marks they need to proceed, your programme will have to implement some decision logic that takes into account both the degree the student is studying on and the marks they have achieved in each course. The progression rules are:

- BSc Computer Science students require:
  - at least a D in F27SA, F27ID and F17LP
  - at least an E in F27PX
- BSc Computer Systems students require:
  - at least a D in F27SA and F27ID
  - at least an E in F27PX and F17LP

You also need to know that:

- Grade D is awarded for a mark greater than or equal to 40% and less than 50%.
- Grade E is awarded for a mark greater than or equal to 30% and less than 40%.

The following are some examples of what the behaviour of your programme should look like.

```
Which degree programme are you studying on? (Enter CSci or CSys)
CSci
Please enter marks for the courses F27SA, F27ID, F27PX and F17LP
75 62 54 40
Your average mark across these four courses is 57.75%
For this semester, you have met the requirements to proceed to your
next year of study for the degree programme BSc Computer Science.
```

```
Which degree programme are you studying on? (Enter CSci or CSys)
CSci
Please enter marks for the courses F27SA, F27ID, F27PX and F17LP
59 73 56 34
Your average mark across these four courses is 55.5%
For this semester, you have not yet met the requirements to proceed
to your next year of study for the degree programme BSc Computer
Science.
```

(continues on next page…)

Note that you are **not** required to tell a student *why* they haven't yet met the progression criteria.

Interface Requirements

To get full marks, your user interface (i.e. how you ask the user for input and how you format the output) needs to meet the following requirements:

- The user should be able to indicate which degree programme they are studying on, and what marks they have in each of the four courses. You don't have to read all the marks in on the same line, as shown in the examples.
- The user should be able to enter an abbreviated form of the degree name, e.g., CSci and CSys as in the examples. This is to save the user from typing out the full name.
- The output should be formatted as in the examples. This includes showing the average mark as a floating-point value with a percentage symbol after it, and showing the full name of the degree programme.

The program is not required to do any error checking of input provided by the user, so for the purpose of this assessment you can assume that the user will always provide input that is formatted in an appropriate way. However, you will not be penalised if you do implement error checking.

Coding Requirements

The following are requirements and guidance for how you should implement the program. Read these carefully, because you may lose marks if you don't follow them:

- Course marks are always whole numbers, so the marks entered by the user should be stored in integer variables.
  - Remember to use meaningful variable names that follow Java's naming conventions.
- To work out the average mark, you need to add up the four course marks and divide by four.
  - Be aware of operator precedence; if unsure, use extra parentheses to enclose sub-expressions.
  - Remember that dividing an integer by an integer results in an integer.
- You should use constants to specify the lower bounds of the D and E grades, and then use these constants when comparing marks within your code.
  - i.e. rather than having code like `if(F27SA<40)`, it should instead be `if(F27SA<D)`
- It is important to use the correct indentation when using `if…else` statements; otherwise, your code will be hard to read. Look at the examples in the lectures if you're unsure.

Commit your changes to GitLab every time you get something working. If you later break something, this will make it easy for you to restore the earlier working version. Once you've finished, **make sure you commit your code to GitLab** before you get it marked. If you have not been able to finish everything, please commit what you have been able to do.

Marks

You will get a mark out of 10 for your work. Here is a guide to how your work will be marked, though the final mark is up to the person marking your work, and will reflect the understanding of the course materials shown in your work**:**

- 1 mark for using GitLab correctly, i.e. forking, checking out the code stub, and committing
- 2 marks if your code is well-formatted (including correct indentation), is appropriately commented, and uses sensible names for variables that follow Java's naming conventions
- 2 marks for correctly using variables and constants, and reading in user input

(continues on next page…)

- 2 marks for correctly calculating the course average and displaying this to the user in the appropriate format
- 3 marks for correctly implementing the decision logic and informing the user of whether they have met the progression requirements

Late submissions will be marked according to the university's late submissions policy, i.e. a 30% deduction if submitted within 5 working days of the original deadline, and no mark after that.

If you have mitigating circumstances (e.g. illness), please submit a mitigating circumstances application, as described at:

https://www.hw.ac.uk/students/studies/examinations/mitigating-circumstances.htm