CSP1150/CSP5110 Programming Principles

Assignment 1:	Individual programming assignment ("Student Grades" program)
Assignment Marks:	Marked out of 20, worth 20% of unit
Due Date:	09 April 2018, 9:00AM

Background Information

This assignment tests your understanding of and ability to apply the programming concepts we have covered in the unit so far, including the usage of variables, input and output, data types, selection, iteration, functions and data structures. Above all else, it tests *your ability to design and then implement a solution to a problem* using these concepts.

Assignment Overview

You are required to design and implement a "Student Grades" program that allows the user to input the assessment marks for students in a class. The program uses this information to determine and display information such as student grades (e.g. "Distinction"). The program displays information on both a per-student / per-assessment basis, as well as some aggregated data such as a class average.

Details of the program requirements can be found in the "Program Requirements" section.

The entirety of this program can be implemented in approximately 100 lines of code (although implementing CSP5110 requirements or optional additions may result in a program longer than this). Ask your tutor for advice if you feel your program is unusually long or inefficient.

Pseudocode

As emphasised by the case study of Module 5, it is important to take the time to properly *design* a solution before starting to write code. Hence, this assignment requires you to *write and submit pseudocode of your program design* as well as the code for the program. Furthermore, while your tutors are happy to provide help and feedback on your assignment work throughout the semester, they will expect you to be able to show your pseudocode and explain the design of your code.

You will gain a lot more benefit from pseudocode if you actually attempt it *before* trying to code your program – even if you just start with a rough draft to establish the overall program structure, and then revise and refine it as you work on the code. This back and forth cycle of designing and coding is completely normal and expected, particularly when you are new to programming. The requirements detailed on the following pages should give you a good idea of the structure of the program, allowing you to make a start on designing your solution in pseudocode.

See Reading 3.3 for further information and tips regarding writing good pseudocode.

To help you get started with your program design, a broad overview has been provided below:

```
Prompt user for number of assessments
Prompt user for assessment names and values

If sum of assessment values is not 100
    Show error message
Else
    Prompt user for number of students

    For each student
        Prompt user for student name

        For each assessment
            Prompt user for assessment mark
            Calculate and display assessment grade

        Calculate and display student's total mark and grade

    Calculate and display class average and top student details
```

This is an *extremely general overview* that only aims to reflect the *general structure* of the code and the *main steps of processing* involved. Your pseudocode should be significantly more detailed. As always, contact your tutor if you have questions or do not understand any part of the assignment, or if you would like help or feedback on your work.

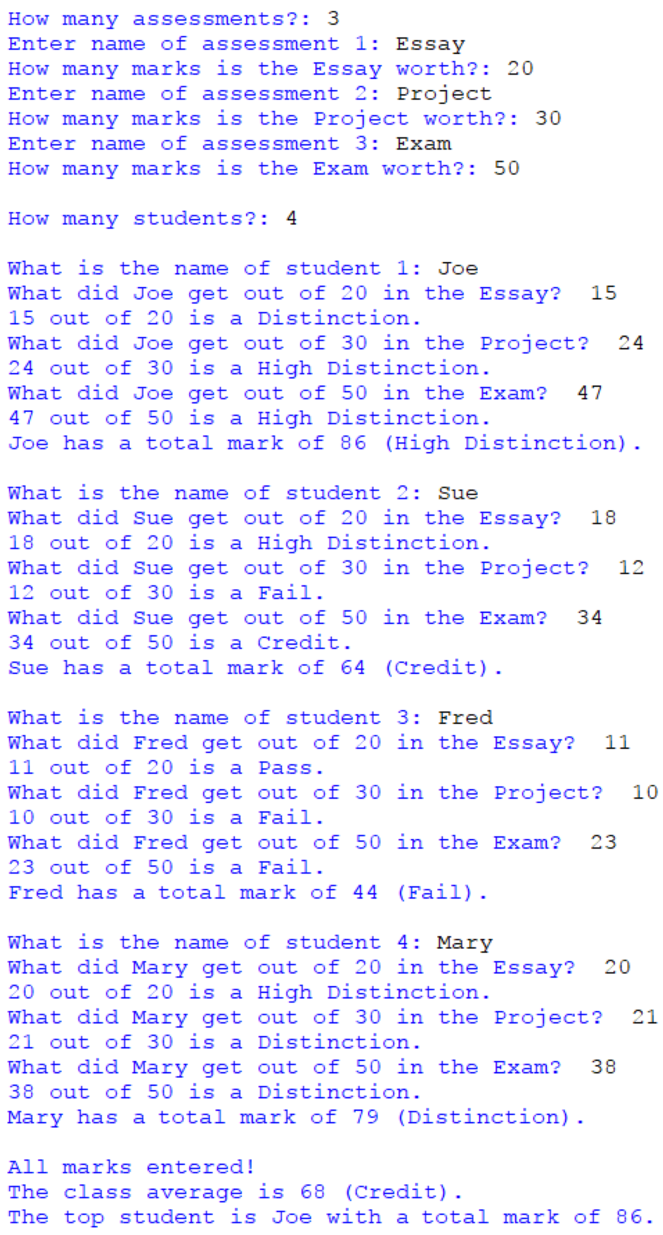
Write a *separate section of pseudocode for each function* you define in your program so that the pseudocode for the main part of your program is not cluttered with function definitions. Ensure that the pseudocode for each of your functions clearly describes the parameters that the function receives and what the function returns back to the program.

It may help to think of the pseudocode of your program as the content of a book, and the pseudocode of functions as its appendices: It should be possible to read and understand a book without necessarily reading the appendices, however they are there for further reference if needed.

Only one function is required in this assignment (detailed later in the assignment brief).

Program Output Example

To help you to visualise the program, here is an example screenshot of the program being used:



```
How many assessments?: 3
Enter name of assessment 1: Essay
How many marks is the Essay worth?: 20
Enter name of assessment 2: Project
How many marks is the Project worth?: 30
Enter name of assessment 3: Exam
How many marks is the Exam worth?: 50

How many students?: 4

What is the name of student 1: Joe
What did Joe get out of 20 in the Essay?  15
15 out of 20 is a Distinction.
What did Joe get out of 30 in the Project?  24
24 out of 30 is a High Distinction.
What did Joe get out of 50 in the Exam?  47
47 out of 50 is a High Distinction.
Joe has a total mark of 86 (High Distinction).

What is the name of student 2: Sue
What did Sue get out of 20 in the Essay?  18
18 out of 20 is a High Distinction.
What did Sue get out of 30 in the Project?  12
12 out of 30 is a Fail.
What did Sue get out of 50 in the Exam?  34
34 out of 50 is a Credit.
Sue has a total mark of 64 (Credit).

What is the name of student 3: Fred
What did Fred get out of 20 in the Essay?  11
11 out of 20 is a Pass.
What did Fred get out of 30 in the Project?  10
10 out of 30 is a Fail.
What did Fred get out of 50 in the Exam?  23
23 out of 50 is a Fail.
Fred has a total mark of 44 (Fail).

What is the name of student 4: Mary
What did Mary get out of 20 in the Essay?  20
20 out of 20 is a High Distinction.
What did Mary get out of 30 in the Project?  21
21 out of 30 is a Distinction.
What did Mary get out of 50 in the Exam?  38
38 out of 50 is a Distinction.
Mary has a total mark of 79 (Distinction).

All marks entered!
The class average is 68 (Credit).
The top student is Joe with a total mark of 86.
```

The program prompts for a number of assessments, and then prompts for the name and value of that many assessments.

Since the values add up to 100, the program continues – prompting for a number of students.

The program prompts for the name of the first student, followed by the mark for each assessment. The corresponding grade is shown for each mark, as well as showing the total mark and final grade.

The process repeats for the second student…

And the third…

And the fourth…

Once all of the students' marks have been entered, the program shows the class average mark and corresponding grade, and who the top student is.

Program Requirements

Below are the detailed requirements of the program you must implement. In the following information, numbered points describe a *core requirement* of the program, and bullet points (in italics) are *additional details, notes and hints* regarding the requirement. Refer to the earlier information or ask your tutor if you do not understand a requirement or would like further details.

1. Prompt the user to input the number of assessments that the unit has.

- *This value will be referred to as "`numAssessments`" in this assignment brief.*
- *Remember to convert numeric inputs such as this to integers so that you can use them as intended!*

2. For each assessment, prompt the user to input a name (e.g. "Essay", "Exam"…) and value (how many marks it is worth). Store these values in two separate lists – a list of assessment names (strings) and another list of assessment values (integers).

- *Use a loop that repeats `numAssessments` times. Prompt for the name and value and append them to the lists inside the body of the loop.*

3. After the user has entered the name and value for all assessments, if the assessment values do not add up to 100 the program should show an error message and end. If the assessment values DO add up to 100, prompt the user to enter the number of students in the class.

- *This value will be referred to as "`numStudents`" in this assignment brief.*

4. Enter a loop that repeats `numStudents` times. The body of this loop must:

4.1. Prompt the user to type the student's name and store it into a variable.

4.2. Enter a loop that repeats `numAssessments` times. The body of this loop must:

- *Note that the loop for assessments is <u>inside the body of the loop for students</u> – this is how the program can prompt for the appropriate number of assessments per student.*

4.2.1. Prompt the user to input the current student's mark for the current assessment.

- *The prompt for input should include the current student's name, as well as the name and value of the current assessment, e.g. "What did Joe get out of 20 in the Essay?"*

4.2.2. If the mark entered by the user is below 0, set it to 0. If the mark is above the value of the current assessment, set it to the value of the current assessment.

- *e.g. If the user enters a mark of 30 for an assessment with a value of 20, change the mark to 20. This is simply to ensure that the mark is valid – alternatively, you can use a loop to re-prompt the user for input until they provide a valid mark.*

4.2.3. Determine and print the grade corresponding to the assessment mark by using the "`calculateGrade`" function (detailed later in this assignment brief).

- *The "`calculateGrade`" function expects a percentage out of 100, so be sure to convert the mark appropriately using the assessment's value, e.g. 13 out of 20 converts to 65%.*
- *The message printed should include the mark, assessment value, and the grade, e.g. "13 out of 20 is a Credit."*

4.3. After obtaining marks for all assessments for a student, the program must determine the student's total mark and display a message which includes their name and their final grade, e.g. "Joe has a total mark of 72 (Distinction)".

- *This occurs inside the body of the loop that repeats once per student, but after (and not part of) the loop that repeats once per assessment. Refer to the overview in the Pseudocode section.*
- *To determine a student's final grade, add up all of their assessment marks then use the "`calculateGrade`" function to determine the grade.*
- *Your program will need to keep track of the student's assessment marks. Creating a variable that is set to 0 at the start of the student loop and then has marks added to it in the assessment loop is one approach (i.e. a running total). See Workshop 3, Task 4 for a similar task.*

5. The last thing the program should (after looping through all of the students) do determine and print the average mark (and corresponding grade) for the class as a whole, and the name and total mark of the top student – i.e. the student with the highest total mark.

- *It is up to you to figure out how to calculate these values. It will involve keeping track of the information necessary throughout earlier parts of the program.*

Programming Tip: Do not attempt to implement the entire program at once. Work on one small part (a single requirement or even just *part* of a requirement) and only continue to the next part once you have made sure that the previous part is working as intended and that you understand it.

It can also be useful to create separate little programs to work on or test small sections of complex code, allowing you to focus on just that part without the rest of the program getting in the way.

The "calculateGrade" function

Your program needs to determine the grade corresponding to a mark in three different parts of the program – when displaying a student's grade for a single assessment (Requirement 4.2.3), when displaying a student's final grade (Requirement 4.3) and when displaying the average grade for the class (Requirement 5). Therefore it makes sense to define a function that can be sent a mark as a parameter and returns a string of the corresponding grade, e.g. sending 56 should return "Pass".

Your program must define a function named "calculateGrade" that receives a mark out of 100 as a parameter. It must determine and return the corresponding grade using the thresholds in this table:

Mark	Grade
0 to 49	Fail
50 to 59	Pass
60 to 69	Credit
70 to 79	Distinction
80 to 100	High Distinction

The definition of the function should be at the start of the program, and it should be used where needed in the program. Revise Module 4 if you are uncertain about defining and using functions, and be sure to implement it so that it receives and returns values exactly as described above.

Additional Requirements for CSP5110 Students

If you are in CSP5110, the following additional requirements apply. If you are in CSP1150, you do not need to do implement these requirements (but you are encouraged to do so if you want). Ask your tutor if you do not understand any of the requirements or would like further information.

1. Include code to allow the user to simply press Enter without typing anything when prompted to enter a student's assessment mark. Doing so should be counted as entering a value of 0.

2. As well as determining and printing the class average and top student information at the end of the program (Requirement 5), the program must display the number and percentage of students who passed the class (total mark of at least 50), e.g. "25/30 students (83%) passed the class."

Optional Additions and Enhancements

Below are some suggestions for minor additions and enhancements that you can make to the program to further test and demonstrate your programming ability. They are *not required* and you can earn full marks in the assignment without implementing them.

- Make sure that the output printed by the program is always well presented and formatted. Avoid undesirable extra spaces when concatenating strings and variables, and include extra line breaks where desired to make the program's output easier to read and interpret.

- Include code to check that the user enters a number that is at least 1 when they are prompted to enter the number of assessments, number of students and assessment values. If they enter a value that is less than 1, show an error message and either end the program or continue to re-prompt them until they enter a valid value.

- Ensure that your program does not crash if the user enters something that is not an integer when prompted to enter the number of assessments, number of students, assessment values or marks. This is best done using exception handling, which is covered in Module 6, however what you need to know (and a helpful function to do it) is covered in Workshop 4.

- Adjust the code that determines the top student to make it so that if multiple students have equal highest marks, all of their names are shown at the end of the program.

- When displaying the class average mark at the end of the program, round it to the nearest integer and show it without decimal places, e.g. show "68" instead of "68.0" or "68.125".

Submission of Deliverables

Once your assignment is complete, submit both your **pseudocode** (PDF format) and **source code** (".py" file) to the appropriate location in the Assessments area of Blackboard. An assignment cover sheet is not required, but be sure to **include your name and student number at the top of both files (not just in the filenames)**.

Referencing, Plagiarism and Collusion

The entirety of your assignment **must be your own work** (unless otherwise referenced) and produced for the current instance of the unit. Any use of unreferenced content you did not create constitutes plagiarism, and is deemed an act of academic misconduct. All assignments will be submitted to plagiarism checking software which includes previous copies of the assignment, and the work submitted by all other students in the unit.

Remember that this is an **individual** assignment. *Never give anyone any part of your assignment – even after the due date or after results have been released. Do not work together with other students on individual assignments – helping someone by explaining a concept or directing them to the relevant resources is fine, but doing the assignment for them or alongside them, or showing them your code is not appropriate.* An unacceptable level of cooperation between students on an assignment is collusion, and is deemed an act of academic misconduct. If you are uncertain about plagiarism, collusion or referencing, simply contact your tutor, lecturer or unit coordinator and ask.

You may be asked to **explain and demonstrate your understanding** of the work you have submitted.

Marking Key

Marks are allocated as follows for this assignment.

Criteria	Marks
Pseudocode These marks are awarded for submitting pseudocode which suitably represents the design of your source code. Pseudocode will be assessed on the basis of whether it clearly describes the steps of the program in English, and whether the program is well structured.	**5**
Functionality These marks are awarded for submitting source code that implements the requirements specified in this brief, in Python 3. Code which is not functional or contains syntax errors will lose marks, as will failing to implement requirements as specified.	**10**
Code Quality These marks are awarded for submitting well-written source code that is efficient, well-formatted and demonstrates a solid understanding of the concepts involved. This includes appropriate use of commenting and adhering to best practise.	**5**

Total:	**20**