4COSC001W: Programming Principles I - Assignment Specification (2019/20)	
Module leader	W Purdy
Unit	Practical Exercises
Weighting:	50%
Qualifying mark:	30%
Description:	Practical Work
Learning Outcomes Covered in this Assignment:	The coursework rationale is: • LO1 Identify program requirements and select appropriate algorithms to implement them; • LO2 Represent algorithms in a structured manner (e.g., by use of flow diagrams or pseudo-code); • LO3 Implement algorithms using an algorithmic, strongly typed programming language, and design and run appropriate tests on the resulting code; • LO4 Write program code that conforms to norms of good style and meets generally accepted referencing criteria;
Handed Out:	21st October 2019
Due Date:	**Tuesday 19th November at 1:00 PM**
Expected deliverables:	a) Flow chart (Part 1 and Part 2) b) Python program code - **Important:** Submit your python code file created in IDLE using the name convention: "student_id.py", e.g. w1234567.py - **DO NOT** submit your code as word, notepad or a PDF document. c) Test case results (Part 1 and Part 2) d) Demo - Monday 25 November (during your scheduled tutorial)
Method of Submission:	Submitted online via Blackboard
Type of Feedback and Due Date:	Written feedback and marks 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.
BCS Criteria covered in this Assignment:	2.1.1 Knowledge and understanding of facts, concepts, principles & theories 2.1.2 Use of such knowledge in modelling and design 2.1.3 Problem solving strategies 2.2.1 Specify, design or construct computer-based systems 2.2.4 Deploy tools effectively 2.3.2 Development of general transferable skills 4.1.1 Knowledge and understanding of scientific and engineering principles 4.1.2 Knowledge and understanding of mathematical and statistical principles

Assessment regulations
Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission
If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website: **http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description

The University requires a program to predict progression outcomes at the end of each academic year. You should write this program in Python using the data shown in Table 1.

Part 1 - Student Version

1. The program should allow students to predict their progression outcome at the end of each academic year.
2. The program should prompt for the number of credits at pass, defer and fail and then display the appropriate progression outcome for an individual student.
3. The program should let the user know if a credit input is the wrong data type. I.e., 'Integers required' is displayed.
4. The program should let the user know if credits are not in the range 0, 20, 40, 60, 80, 100 and 120. I.e., 'Range error' is displayed.
5. The program should let the user know if the total of the pass, defer and fail credits is not 120. I.e., 'Total incorrect' is displayed.

- Use conditions and user-defined functions in your solution as appropriate.
- Flow Diagram - Before you start to program your solution you should create your flow diagram that represents your algorithm in a structured manner. Submit flowchart for Part 1 for marking.
- Test Plan - A Part 1 test plan is provided in the appendix. Submit the completed test plan (with your flow diagram and program code) and bring a printed copy of the test plan to the demo.

Table 1: Progression outcomes as defined by the University regulations.

	Volume of Credit at Each Level			Progression Outcome
	Pass (including condoned pass)	Defer	Fail	
1	120	0	0	Progress
2	100	20	0	Progress – module trailer
3	100	0	20	Progress – module trailer
4	80	40	0	Do not Progress – module retriever
5	80	20	20	Do not Progress – module retriever
6	80	0	40	Do not Progress – module retriever
7	60	60	0	Do not progress – module retriever
8	60	40	20	Do not progress – module retriever
9	60	20	40	Do not progress – module retriever
10	60	0	60	Do not progress – module retriever
11	40	80	0	Do not progress – module retriever
12	40	60	20	Do not progress – module retriever
13	40	40	40	Do not progress – module retriever
14	40	20	60	Do not progress – module retriever
15	40	0	80	Exclude
16	20	100	0	Do not progress – module retriever
17	20	80	20	Do not progress – module retriever
18	20	60	40	Do not progress – module retriever
19	20	40	60	Do not progress – module retriever
20	20	20	80	Exclude
21	20	0	100	Exclude
22	0	120	0	Do not progress – module retriever
23	0	100	20	Do not progress – module retriever
24	0	80	40	Do not progress – module retriever
25	0	60	60	Do not progress – module retriever
26	0	40	80	Exclude
27	0	20	100	Exclude
28	0	0	120	Exclude

Part 2 - Staff Version

This extension should meet the requirements specified for Part 1 but also allow a staff member to predict progression outcomes for multiple students.

1. The program should prompt for credits at pass, defer and fail and display the appropriate progression for each individual student until the staff member user enters 'q' to quit.
2. When 'q' is entered, the program should produce a 'histogram' where each star represents a student who achieved a progress outcome in the category range: progress, trailing, module retriever and exclude. See example below.
3. The program should display the number of students for each progression category and the total number of outcomes processed.

- The program will make use of loops and user-defined functions.
- Flow Diagram - Before you start to program your solution you should create your flow diagram that represents your algorithm in a structured manner. Submit flowchart for Part 2 for marking.
- Test Plan – You are required to create your own test plan for Part 2. Submit the completed Part 2 test plan (with your flow diagram and program code) and bring a printed copy to the demo.

This following horizontal histogram example shows the output distribution for 20 outcomes. However, your program should work with any number of outcomes generated.

```
Progress 10:  **********
Trailing 5:     *****
Retriever 3:   ***
Excluded 2:    **

20 outcomes in total.
```

Part 3 - Vertical Histogram (optional extension)

Extend your program to add an additional histogram that displays vertically so the stars in a category should go downwards and not across the screen, e.g.:

```
Progress  Trailing  Retriever  Excluded
   *         *          *          *
             *          *
             *
```

- Hint: as a line is printed decide if each category needs a star or space.
- Part 3 does NOT require a flow diagram or a test plan.

Part 4 – Alternative Staff Version (optional extension)

For this staff version the data will be accessed from a list, tuple or dictionary and NOT from user input. The data held in the list, tuple or dictionary will match the test cases 1 to 10 shown in the appendix. Use user-defined functions.

- Part 4 does NOT require a flow diagram or a test plan.
- If attempted, the code for **both** staff versions (Part 2 and Part 4) must be submitted for marking.

Programming Style & References

Use descriptive variable names and reference any code taken from others sources in your program code. Include the following at the top of your program(s).

```
# I declare that my work contains no examples of misconduct, such as plagiarism, or collusion.
# Any code taken from other sources is referenced within my code solution.

# Student ID: ……..

# Date:  ……..
```

Coursework Demo

Demonstrate your working solution to your tutor during your scheduled tutorial. Marks will be allocated for your ability to answer questions relating to your program. Bring a printed copy of your submitted test cases to the demonstration. **NOTE: If you do not attend your demo your Coursework mark will be capped at 30 marks.**

Marking scheme

The Coursework will be marked based on the following marking criteria:

Criteria	Max. Marks per subcomponent	Mark per component
Assignment - Progress Outcomes		
Part 1 - Student Version (20) • Credits entered & progress outcome displayed. • Use of conditional statements. • Catches input that is the wrong data type • Credits outside range: 0, 20, 40, 60, 80, 100, 120 • Credit total not 120 • User-defined functions Part 1 Flow chart (matches submitted program) (9) Part 1 Test plan in Appendix used. Marks allocated where test is PASS & matches submission (7)	36	36
Part 2 - Staff Version (18) • Predict progression outcomes for multiple students. • User enters 'q' to quit. • 'histogram' for progress outcomes (progress, trailing, module retriever and exclude) • Category totals and overall total displayed • User-defined functions Part 2 Flow chart (matches submitted program) (9) Part 2 Student creates own Test Plan (8)	35	71
Part 3 - Vertical Histogram (optional extension)	4	75
Part 4 - Alternative Staff Version (optional extension)	6	81
Programming style	2	83

Demo – Marks allocated for your ability to answer questions and demonstrate understanding of your solutions.
• If you are cannot explain you code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then no marks will be given for the demo of that component.
  • Part 1 (7)
  • Part 2 (6)
  • Part 3 (2)
  • Part 4 (2)
	17	

NOTE: If you do not attend your demo your **overall coursework mark will be capped at 30 marks**		
Total:		100

Submit this completed test plan with your solution code and flow chart				
Student Name:			**Student ID:**	
Test No.	Test Input	Expected Result	Actual Result (State if not attempted)	Pass/Fail
1	Pass = 120	'Progress' is displayed		
2	Pass = 100 Defer = 20	'Progress – module trailer' is displayed		
3	Pass = 100 Fail = 20	'Progress – module trailer' is displayed		
4	Pass = 80 Defer = 20 Fail = 20	'Do not Progress – module retriever' is displayed		
5	Pass = 60 Defer = 40 Fail = 20	'Do not Progress – module retriever' is displayed		
6	Pass = 40 Defer = 40 Fail = 40	'Do not Progress – module retriever' is displayed		
7	Pass = 20 Defer = 40 Fail = 60	'Do not Progress – module retriever' is displayed		
8	Pass = 20 Defer = 20 Fail = 80	'Exclude' is displayed		
9	Pass = 20 Fail = 100	'Exclude' is displayed		
10	Fail = 120	'Exclude' is displayed		
Checking input data type, credit range and credit total:				
11	Pass = a	'Integers required' displayed		
12	Defer = b	'Integers required' displayed		
13	Fail = c	'Integers required' displayed		
14	Pass = 2	'Range error' is displayed		
15	Defer = 5	'Range error' is displayed		
16	Fail = 10	'Range error' is displayed		
17	Pass = 100 Defer = 40	'Total incorrect' is displayed		
18	Pass = 60 Fail = 40	'Total incorrect' is displayed		