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.

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

	Volume of Credit at Each Level			Progression Outcome
	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 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 (i.e., progress, trailing, module retriever or exclude).

   Part 1 - An example of the program running with user input (shown in bold) :

```
------------------------------------------------------------
Please enter your credits at pass: 100
Please enter your credit at defer: 0
Please enter your credit at fail: 20
Progress (module trailer)
------------------------------------------------------------
```

- Marks will be allocated for the efficient use of conditional statements.

- Submit the completed part 1 test plan (in appendix) with your final part 1 solution.

Part 2 – Student Version (Validation)

Extend the Student Version to add the following validation.

1. The program should display '**Integer required**' if a credit input is the wrong data type.
2. The program should display '**Out of range**' if credits entered are not in the range 0, 20, 40, 60, 80, 100 and 120.
3. The program should display '**Total incorrect**' if the total of the pass, defer and fail credits is not 120.
4. The program could loop until acceptable inputs are entered. However, this is optional and will not be allocated a mark.

   Part 2 - An example of the program running with user input (shown in bold):

```
------------------------------------------------------------
Please enter your credits at pass: p
Integer required

Please enter your credits at pass: 140
Out of range.

Please enter your credits at pass: 100
Please enter your credit at defer: 40
Please enter your credit at fail: 20
Total incorrect.

Please enter your credits at pass: 100
Please enter your credit at defer: 20
Please enter your credit at fail: 0
Progress (module trailer)
------------------------------------------------------------
```

- Use user-defined functions in your solution as appropriate.
- Submit the completed part 2 test plan (in appendix) with your final part 2 solution.

Part 3 - Staff Version with Histogram

This extension should meet the requirements specified for Part 1 and 2 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. Optionally you can use an input of **'y' to continue.**
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. **The histogram should relate to the data input entered by the staff member during the program run and work for any number of outcomes**.
3. Display the number of students for each progression category and the total number of students.

   Part 3 - Example of a program run and input (in bold). Note: program should exit on 'q' to quit. 'y' to continue shown in the example is optional and depends on your program structure.

```
------------------------------------------------------------
Staff Version with Histogram

Enter your total PASS credits: 120
Enter your total DEFER credits: 0
```

```
Enter your total FAIL credits: 0
Progress

Would you like to enter another set of data?
Enter 'y' for yes or 'q' to quit and view results: y

Enter your total PASS credits: 100
Enter your total DEFER credits: 0
Enter your total FAIL credits: 20
Progress (module trailer)

Would you like to enter another set of data?
Enter 'y' for yes or 'q' to quit and view results: y

Enter your total PASS credits: 80
Enter your total DEFER credits: 20
Enter your total FAIL credits: 20
Do not progress – module retriever

Would you like to enter another set of data?
Enter 'y' for yes or 'q' to quit and view results: y

Enter your total PASS credits: 60
Enter your total DEFER credits: 0
Enter your total FAIL credits: 60
Do not progress – module retriever

Would you like to enter another set of data?
Enter 'y' for yes or 'q' to quit and view results: y

Enter your total PASS credits: 40
Enter your total DEFER credits: 0
Enter your total FAIL credits: 80
Exclude

Would you like to enter another set of data?
Enter 'y' for yes or 'q' to quit and view results: q

------------------------------------------------------------
Horizontal Histogram
Progress 1  : *
Trailer 1   : *
Retriever 2 : **
Excluded 1  : *

5 outcomes in total.
------------------------------------------------------------
```

- The program will make use of loops and user-defined functions.
- Submit the completed part 3 test plan (in appendix) with your final part 3 solution.

Part 4 - Vertical Histogram (optional extension)

Extend your program to add a vertical histogram (stars in a category should go downwards), e.g.;

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

- Hint: as a line is printed decide if each category needs a star or space.
- If attempted, the code for **both** staff versions (Part 3 and Part 4) must be submitted for marking.

- Submit the completed part 4 test plan (in appendix) with your final part 4 solution.

Part 5 – Alternative Staff Version (optional extension)

- For this version the data will be obtained from a list, tuple or dictionary and NOT from user input. **Hint:** you could use a two-dimensional list (a list of lists) to hold the data.
- A histogram related to the data stored in the list, tuple or dictionary should be displayed.
- The data to use is shown in the table below.
- The solution should use user-defined functions.
- Submit the completed part 5 test plan (in appendix) with your final part 5 solution.

Data to use for Part 5
Pass = 120, Defer = 0, Fail = 0
Pass = 100, Defer = 20, Fail = 0
Pass = 100, Defer = 0, Fail = 20
Pass = 80, Defer = 20, Fail = 20
Pass = 60, Defer = 40, Fail = 20
Pass = 40, Defer = 40, Fail = 40
Pass = 20, Defer = 40, Fail = 60
Pass = 20, Defer = 20, Fail = 80
Pass = 20, Defer = 0, Fail = 100
Pass = 0, Defer = 0, Fail = 120

Part 5 - Example of a program run (no user input used).

```
Progress
Progress (module trailer)
Progress (module trailer)
Do not Progress - module retriever
Do not Progress - module retriever
Do not Progress - module retriever
Do not Progress - module retriever
Exclude
Exclude
Exclude

Progress  1: *
Trailing  2: **
Retriever 4: ****
Excluded  3: ***

10 outcomes in total.
```