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.

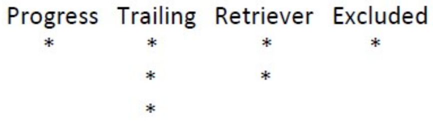
This following horizontal histogram example shows the output distribution for 20 outcomes. However, your program should work with any number of outcomes generated.

```
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.