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

	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

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

Part 1 - Main Version

A. Outcomes

- The program should allow students to predict their progression outcome at the end of each academic year. 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).

B. Validation

- The program should display 'Integer required' if a credit input is the wrong data type.
- The program should display 'Out of range' if credits entered are not in the range 0, 20, 40, 60, 80, 100 and 120.
- The program should display 'Total incorrect' if the total of the pass, defer and fail credits is not 120.
- A few marks will be allocated for the efficient use of conditional statements. For example, the program does not need 28 conditional statements for 28 outcomes.
- 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)
```

C. Multiple Outcomes

- The program loops to allow a staff member to predict progression outcomes for multiple students.
- The program should prompt for credits at pass, defer and fail and display the appropriate progression for each individual student until the staff member enters 'q' to quit. Optionally you can use an input of 'y' to continue.
- See example of program run combined with Histogram below.

D. Histogram

- 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 during the program run and work for any number of outcomes.
- Display the number of students for each progression category and the total number of students.
- 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.

Example Output:

```
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
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
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


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

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

Part 2 – List (extension) (6 marks)

For Part 1, most of the solutions would use variables to store the input data. For Part 2, extend your solution so that the program saves the input progression data to a list or nested list. Then access the stored data from the list and print the data in the following format below.

Example Output: The following should display after the histogram

```
Part 2:
Progress - 120, 0, 0
Progress (module trailer) - 100, 0, 20
Module retriever - 80, 20, 20
Module retriever - 60, 0, 60
Exclude – 40, 0, 80
```

Part 3 - Text File (extension) (6 marks)

For this part you could create an additional Part 3 program or extending your original version. Use python to save any inputted progression data to a text file. Later in the program, access the stored data and print out as shown below. Example output (with data from text file):

```
Part 3:
Progress - 120, 0, 0
Progress (module trailer) - 100, 0, 20
Module retriever - 80, 20, 20
Module retriever - 60, 0, 60
Exclude – 40, 0, 80
```

Part 4 – Dictionary (separate program) (6 marks)

For Part 4, create a program that saves the input progression data to a dictionary or nested dictionary. Then access the data stored in the dictionary and print to the screen. The solution should collect the unique student ids as part of the input and display with the outcomes. Example output is shown below.

```
Part 4:
w1234567 : Progress - 120, 0, 0 w1234566 : Progress (module trailer)
- 100, 0, 20 w1234565 : Module retriever - 80, 20, 20 w1234564 :
Module retriever - 60, 0, 60 w1234563 : Exclude
– 40, 0, 80
```