Year 11 Computer Science Programming Practical Test

Task Outline

- This is an in class practical assessment task.
- You will have forty (40) minutes to complete this test.
- You are to write a number of programs to solve the problems that have been given.

Instructions

1. Create a folder in your Computer Science folder called **Programming Prac Test**
2. Copy **all** the supplied files from **Nexus** and place them in your **Programming Prac Test** folder.
3. Use the supplied files to complete the following problems.
4. Submit **all** files to **Nexus**. Remember, you will receive some marks for partially working code, so attempt **all** questions.

Notes

- You are able to use the Python Reference Sheets that have been supplied.
- You are **NOT** allowed to access any other websites or any other programs that you have written
- You can assume that all values used for testing will be valid. For example, if the questions requires integer inputs, you can assume all values entered will be integers.

Question 1 (6 marks)

Ellyse loves watching the AFL but since she is only 3 she has trouble working out how many points her team has scored. You have decided to write a program to help her!

In footy, each team scores 6 points for a goal and 1 point for a behind. So, if they kick 5 goals and 5 behinds, they will have a total of 35 points (5 * 6 for the goals plus 5 * 1 for the behinds).

You program should:

- Ask the user to enter the number of goals scored
- Ask the user to enter the number of behinds scored
- Output the total score as a single number (using the wording shown below)

NOTE: You can assume then the input will always be an integer between 0 and 50 (inclusive).

Example 1:

```
Goals: 5
Behinds: 5
Your team scored 35 points.
```

Example 2:

```
Goals: 10
Behinds: 2
Your team scored 62 points.
```

Example 3:

```
Goals: 17
Behinds: 12
Your team scored 114 points.
```

A template has been provided for you in the file **question1.py**. Add your code to this file and submit it to Nexus.

Question 2 (6 marks)

Mr Vaswani wants a quick way to calculate the grades for each student at CCGS and has asked for your help. To calculate a student grade for each subject, Mr Vaswani wants to find the average mark across the entire course and allocate a grade based on the following table:

Average Mark	Grade
>= 80	A
>= 65	B
>= 50	C
< 50	D

You are to write a function called *findGrade*. Your function should:

- Take in a list of marks as a parameter
- Calculate the average mark
- Calculate the grade based on the table above
- Return the letter for the grade

NOTE: You can assume that all marks entered will be an integer between 0 and 100 (inclusive).

Example calls to the function:

```
marks = [80, 90, 85]
print(findGrade(marks))
marks = [20, 52, 38, 48, 55]
print(findGrade(marks))
print(findGrade([56, 60, 67, 47, 55, 61, 64]))
```

Teminal output:

```
A
D
C
```

A template has been provided for you in the file **question2.py**. Add your code to this file and submit it to Nexus.

Question 3 (6 marks)

Jake is learning to write the words for numbers, but still gets confused sometimes. You have decided to help him by writing a function that will work out the words for integers from 0 to 99.

You are to write a function called *number_to_words*. Your function should:

- Take in an integer as a parameter
- Calculate the correct words for the number
- Return the words as a string in lowercase
- If the number entered is not between 0 and 99 (inclusive), return "unrecognised number"

When writing the numbers from 21 to 99 (inclusive) as words, you should hyphenate the words. For example, the number 23 should be written as twenty-three (note the hyphen between the twenty and the three).

NOTE: You can assume that the input to the function will always be an integer.

Example calls to the function:

```
print(number_to_words(0))
print(number_to_words(14))
print(number_to_words(45))
print(number_to_words(99))
print(number_to_words(-6))
```

Terminal output:

```
zero
fourteen
forty-five
ninety-nine
unrecognised number
```

A template has been provided for you in the file **question3.py**. Add your code to this file and submit it to Nexus.

Question 4 (6 marks)

Vikki is the safety officer in charge of the railway bridges that have been built in Western Australia. She has asked you to write a function that will help determine if a train will be able to successfully cross a bridge.

Each bridge will collapse if the total weight of railway cars on it at any one time is greater than a certain weight. In Western Australia, each railway car is 10m long, and each bridge has a length that is a multiple of 10 (you can assume that no partial cars will be on the bridge at any point in time).

The details of each train will be stored in a text file (for example **train1.txt**). Each line will have a single integer that will be the weight of the train car. The cars will cross the bridge in the order they appear in the file.

You are to write a function called *bridge_crossing*. Your function should:

- Take in three parameters:
  - *bridge_length* - the length of the bridge (10 <= L <= 100)
  - *total_weight* - be the total weight that the bridge can hold at any one time (1 <= W <= 10000)
  - *filename* - the name the train details file as a parameter
- Read in the details of the file
- Determine if the train will successfully cross the bridge
- Count the number of railway cars that successfully cross the bridge
- If the train is able to cross the bridge, return the string *Success!*
- If the train will not make it across, return the string *Failure*

Examples using the following text files:

Input file 1 (trains1.txt):

```
50
30
10
10
40
10
```

Input file 2 (trains2.txt):

```
500
300
100
700
100
100
500
```

Example calls to the function:

```
print(bridge_crossing(40, 100, “trains1.txt”))
print(bridge_crossing(30, 1000, “trains2.txt”))
```

Terminal output:

```
Success!
Failure
```

A template has been provided for you in the file **question4.py**. Add your code to this file and submit it to Nexus. Two test files (**trains1.txt** and **trains2.txt**) have been provided for you, although your code may be tested with other files.