19 April 2018 **SH1 - H2 Computing - Lecture Test 1** **Duration: 90 minutes**

This Lecture Test includes a total of 4 questions worth a total of 50 marks.

You must answer all 4 questions.

Please answer all questions by implementing the code using Python 3.

There are 2 Text Files that can be found on the desktop of your computer; the file `MARKS.TXT` **is required to complete Question 3, while the file** `RANGE_DATA.TXT` **is required to complete Question 4.**

You are to create a folder on the desktop of your computer and rename it with your name as specified on your NRIC.

You should copy the 2 Text Files mentioned above into the folder you created.

For each part of each question, save the .py file corresponding to that part in the folder described above. Name the .py file using the question number and part alphabet - e.g., to save the code for Question 1 Part a, use the filename 1a.py.

1a Write an iterative function that takes in 1 parameter, a list my_list, and prints its contents on a single line.

For example, given that my_list is [2, 4, 6], the expected output should be:

2 4 6

Note that you may not call print(my_list), but rather must print each element of the list on the same line, with the elements delimited by a single space.

[2 marks]

1b Write a recursive function that takes in 1 parameter, a string s, and returns the reverse of s.

Note that you may neither use the slice [::-1], nor the built-in functions reverse and reversed.

[2 marks]

1c Write an iterative function that takes in 1 parameter, a non-negative integer n, and returns a list containing all the perfect squares in the range -n to n (inclusive).

Note that an integer n is a perfect square when there exists another integer x such that $x^2 = n$.

[3 marks]

1d Write a recursive function that takes 1 parameter, a positive integer n, and returns a list of all even integers between 0 and n (inclusive) sorted in ascending order.

[3 marks]

2a Write a function that implements the insertion sort algorithm. You may assume that this function takes in 1 parameter, a list, and returns the sorted equivalent of that list.

You may not use the built-in functions, min, max, sort, sorted, index.

[4 marks]

2b Implement a function that takes in 1 parameter, a string s, which stores the binary representation of a non-negative integer, and returns the hexadecimal representation of that value as a string.

Note that a binary representation of a non-negative integer corresponds to the base 2 representation of that value, while a hexadecimal representation corresponds to the base 16 representation of that value.

You may not use the built-in functions bin, oct, hex, or the function int(a, b), where a is a string and b is an integer (though you may use int(c), where c is a string - i.e., you can only use the int() function with 1 parameter).

[6 marks]

3 Write a program that does the following.

- Reads the contents of the file `MARKS.TXT`, which stores the marks obtained by 25 students
- Each line of `MARKS.TXT` stores the Student ID and Mark for each student in the format:

  <Student ID>,<Mark>

  - Note that the data stored in `MARKS.TXT` has already been sorted in descending order
  - Also note that there are no duplicate marks
  - Each Student ID corresponds to an string comprising 6 digits
- Stores the data from the file within a list of 2-tuples, that is:
  `[(id_1, mark_1), (id_2, mark_2), … , (id_n, mark_n)]`
- Prompts the user for a positive integer input between 1 and 100 (inclusive)
  - Proper exception handling must be performed on this input request until an appropriate value is provided
- Using the binary search algorithm, finds and then prints the Student ID whose mark is equal to the specified input
  - Your implementation of the binary search algorithm must:
    - Be written as a function
    - It take in 2 inputs, the list of 2-tuples specified above, and the positive integer that was input by the user
    - Searches for the index of the 2-tuple whose mark value is equal to the positive integer that was input by the user
    - Returns this index if the 2-tuple with that input mark value was found, or else returns -1
- Prints the Student ID of the student with the input mark if a student with that mark was found. For example, suppose the input was 75 and (only) the student with ID 00221 scored 75, then the printed output would be:

```
Student 002211 attained the mark 75.
```

- And if not found, instead prints:

```
No student found that attained the mark 75.
```

[15 marks]

4a Jake runs a shooting range and would like to implement a simple system that would allow his customers to store their performance and determine rankings. He decides on a simple text-based menu interface, and to use a text file to store the performance data.

The initial version of the menu interface is to have 4 options. It should appear as follows:

```
1. Load Data from File
2. Add New Entry
3. Save Data to File
5. Quit

Please select an option [1, 2, 3, 5]:
```

This menu should loop until the option `"5"` is input. In each iteration of the loop, the above should be displayed. If any input aside from `"1"`, `"2"`, `"3"` or `"5"` is entered, an appropriate warning should be displayed before the input request is repeated.

When the option `"1"` is input, the program should load the data in the text-file `RANGE_DATA.TXT`, which contains 1 performance entry per line based on the following format:

<Shooter Name>,<Shot *X*-coordinate>,<Shot *Y*-coordinate>

Where:

- <Shooter Name> is a string corresponding to the name of the shooter that made the shot
- <Shot *X*-coordinate > and <Shot *Y*-coordinate> are both floating point values between -1 and 1 (expressed to 3 decimal places)

Note that the shots are stored in order of occurrence (i.e., in the order they were entered into the system); Jake has indicated that this ordering must be preserved within the file.

Once the file is read the user should be notified. A sample of this is shown below.

```
Please select an option [1, 2, 3, 5]: 1

Data loaded from RANGE_DATA.TXT.
```

When the option `"2"` is input, the user is prompted with the following input queries.

- Username (i.e., the name of the shooter)
- Shot X-Coordinate
- Shot Y-Coordinate

The input values must be validated; they should meet the following criteria:

- Username – string containing at most 20 characters from the English alphabet
- Shot X-Coordinate and Shot Y-Coordinate – floats with exactly 3 decimal places that are greater than -1 and less than 1
- If any input does not meet the above specifications, an appropriate error message should be printed, and the relevant input request repeated.

A sample of the output for Option 2 is as follows.

```
Please select an option [1, 2, 3, 5]: 2

Username: Donald
Shot X-Coordinate: 0.050
Shot Y-Coordinate: -0.025

New shot stored.
```

Note that the above sample output does not match the actual data.

When the option `"3"` is input, `RANGE_DATA.TXT` should be updated to reflect any new data was input using Option 2. Do note that only the data that was input after the last execution of Option 3 should be appended to `RANGE_DATA.TXT`.

If Option 3 is selected when there is no new data, an appropriate warning message should be displayed. Otherwise, if the data was successfully saved, then an appropriate message should be displayed to indicate that the new data was successfully saved.

As mentioned above, when the option `"5"` is input, the program terminates.

It should be noted that Option 2 and Option 3 can only be executed if Option 1 was first executed. This must be enforced within by the system.

Write the program code that implements all the functionality described above.

[7 marks]

4b Finally, Jake would like to include an option to list the best 3 shots made. Add the following option to the menu.

```
4. Display Top 3 Shots
```

When this option (i.e., Option 4) is entered, a sort is performed, ranking all the shots stored based on their distance to the middle of the target.

The following diagram depicts a target and the corresponding coordinates about the target. Each shot is assessed based on its distance to the middle of the target (i.e., to the origin at coordinates (0, 0)).

In order to calculate the distance between 2 points, A, at (x_1, y_1), and B, at (x_2, y_2), we use Pythagoras's theorem:

$$\text{Distance between A and B} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Thus, to find the 3 best shots, you are to:

- Sort all the shots in ascending order based on their distance to the origin
- Display the first 3 shots in the sorted list.

You are to implement the Quicksort algorithm to perform the required sorting.

Each shot listed should include the name of the shooter, and the distance to the centre of the target (rounded to 6 decimal places).

A sample of the output for this option is as follows.

```
Please select an option [1, 2, 3, 4, 5]: 4

Top 3 Shots:

#   Shooter                       Distance to Centre of Target
1   Donald                                            0.123456
2   Barbara                                           0.234567
3   Joseph                                            0.345678
```

Note that the above sample output does not match the actual data.

Write the program code that implements all the functionality described above.

[8 marks]