CE151 COURSEWORK 2 – 2020

Credits: 50% of total module mark. Submission of this assignment will be via FASER (deadline 21th December). You should refer to sections 5 and 7 of the Undergraduate Students' Handbook for details of the University policy regarding late submission and plagiarism; the work handed in must be entirely your own.

Please make separate files for each exercise. Please submit THREE .py files; and name them as: ex1.py, ex2.py and ex3.py. These exercises contribute to **90%** of the mark for coursework2. The remaining **10%** of the total mark for the coursework will be awarded for programming style and documentation. Sample data file to be used for testing the program in **Exercise 3** is supplied [test].

Exercise 1 [20%] Palindromes. A palindrome is a word that is the same backwards and forwards. The word "rotor" is an example. Prepare file **ex1.py**.

(a) Write a function which returns True if [both] two input strings [presenting two words] are palindromes. **[5%]**
(b) Some palindromes ignore spaces and capitalization so "Never odd or even" is an acceptable palindrome. Write another function, that receives a string presenting a sentence and returns True if the input is palindrome. **[10%]**
(c) Write the main program where, first, prompts for two strings [two words] from the user. Then, it calls the function in (a), prints an informative message that whether both strings [both words] are palindrome or not (see (a)). Then, the program needs to prompt for a sentence, call the function in (b), and prints an informative message whether the sentence is palindrome or not (see (b)). **[5%]**

Notes: For part (a), "Rotor" is not considered as a Palindrome word since capitalization is taken into account. However, for part (b), capitalization and spaces are ignored in determination of a Palindrome sentence.

Exercise 2 [20%] Prepare file **ex2.py**.

(a) Write a function that generates two lists [each one to have a length of 15] of random numbers between 0 and 100, sorts them separately, then simply merges the results [concatenate], sorts the final list and prints the final list. **[10%]**
(b) Write a function that takes a string [with no space] as an argument, converts the string to a list of characters, sorts the list, converts the list back to string and prints the resulting string. Assume that the string is in lowercase. **[5%]**
(c) Write your main program and call the two functions above to print the sorted list of random numbers and prompt for the user to provide an input string to print the resulting string. **[5%]**

Exercise 3 [50%] The attributes of a football player are first-name, last-name, position, team and salary. In the file **ex3.py,** you may start by writing a function (tuple-creation function) which takes a string argument, creates and returns a tuple containing details of the player specified by the string. The string should be assumed to have the format

```
Chelsea Right-back Davide Zappacosta 20,000,000
```

The above format has been used in each line of a sample file [test] provided for this assignment. Therefore, the tuple-creation function is useful for reading the file contents and making tuples.

The tuple will contain the fields in the following order: team, position, first-name, last-name and salary. The tuple should contain exactly 5 items.
This exercise involves printing details of all players to the user (see (a)-(f)). As an example, you may need to have another function which prints full details of one player on a single output line. The function should take a tuple as its argument and print the details in fixed-width fields using a layout such as:

```
Zappacosta, Davide 20,000,000 Right-back Chelsea
```

The important point here is about formatting while you print the results e.g. full details of players:

1- You need to follow instructions for the number of characters you allocate for each string as:

- The first-name/last-name/team/position must be displayed in a single fixed-width field (15 characters).
- The salary must contain at most 8 digits; use fixed-width field 10, you need to have it as formatted above using commas for each three digits (e.g 20,000,000).

2. You need to follow instructions for printing which must have this ordering: [last-name, first-name, salary, position, team] as above.

3- To make a neat table [as explained below] or further customize formatting; this is up to you to separate first-name/last-names by comma, to make vertical/horizontal lines, aligning the strings to left/right/centre. Making a neat table is based on your personal preference. The more you enhance your table, the higher chance to get full mark for that.

In the main body of the program **ex3.py**, write your code which prompts the user for a filename and attempts to open the file whose name is supplied. If the file cannot be opened an error message should be printed and the program should terminate; otherwise, the program should read each line from the file and supply it to a player tuple-creation function described above [this an example – use of such function is optional], storing the tuples returned by this function in a list. Please note that creating list of tuples is essential no matter of whichever approach you will take. You may assume that the file contains lines that conform to the format described for the argument to the function.

After all the data in the [test] file has been read, the program should display details of all the players in the list in a neat table **[5%]** (using the functions that are already written), then, enter a loop in which the user should be given the options:

(a) Display full details of the player with a given last-name [user should be prompted to provide the last-name] **[5%]**

(b) Display full details of all players with a salary in a particular range (e.g. 20000 to 30000) [user should be prompted to provide salary range] **[5%]**

(c) The first- and last- names of all players of a team [user should be prompted to provide the team]. **[5%]**

(d) The user should be prompted to supply the position and team. The list should then be searched, and the appropriate output [full details] to be displayed. **[10%]**

(e) The user should be prompted to supply the position, lower and upper bounds of the salary range. The list should then be searched, and the appropriate output to be displayed. The salary output should be sorted numerically in ascending order of salary while [full details] are also provided. **[10%]**

(f) Adding a player to the list (No need to save the results back to file). The user should be prompted to provide the first/last names, team, position and salary. **[10%]**

(g) Quit *Notes: The user should be told what has to be typed in order to select each option; the input options should be concise (e.g. '1', '2' etc. or 'j', 's' etc.) in order to allow quick testing of the program. Please assume that given last-names are unique. Appropriate messages should be displayed if a search produces no results. You lose marks if the options are not written in a loop.* You are allowed to use any built-in function or library for all exercises. All printings in all exercises need to be done in Python shell. No printing to a file is expected.