Lab 5 (Nested List and Dictionary)

1. (Nested list) In a diving competition, every diver makes 3 dive attempts. Each dive attempt is awarded a score which is a value between 0 and 10, inclusive of 0 and 10. The diver with the best total wins.

   You are given the scores for 6 divers:

   ```
   scores=[[7.9,7.8,8.2],[8.0,8.5,8.4],[9.0,9.1,9.5],
               [9.0,9.2,9.2],[8.5,8.8,9.0],[8.7,8.8,8.7]]
   ```

   a. Display the results in the following format:

   ```
   Diver A1  A2  A3  Total
      1  7.9 7.8 8.2 23.9
      2  8.0 8.5 8.4 24.9
      3  9.0 9.1 9.5 27.6
      4  9.0 9.2 9.2 27.4
      5  8.5 8.8 9.0 26.3
      6  9.7 9.8 9.7 29.2
   ```

   You may assume that every diver will make 3 attempts (fixed). However, the number of divers can vary.

   b. Display the top 3 positions in descending order of the total score as follows:

   ```
   Top three positions
   Diver Total
      6  29.2
      3  27.6
      4  27.4
   ```

   Assume there are no ties.

   c. Consider the case where the list is structured as follows:

   ```
   scores = [ [7.9,8.0,9.0,9.0,8.5,9.7],[7.8,8.5,9.1,9.2,8.8,9.8],
               [8.2,8.4,9.5,9.2,9.0,9.7] ]
   ```

   where the list consists of 3 attempts and for each attempt, the score for each diver. Produce the same output as part a.

2. (Nested list) A nested list is used to represent the scores of 2 players A and B in a badminton game as follows:

   ```
   gameScore=[['A','B'],[21,11],[19,21],[20,21]]
   ```

   The above represents 3 games played between player A and B. Based on the game score, the score of the first game score is 21-11 in which Player A is the winner, etc. The overall game score is 1-2 and player B is the winner.

   It is possible that only 2 games are played with the game score, as shown in the following example:

   ```
   gameScore=[['A','B'],[21,1],[21,10]]
   ```

In this example, the overall game score is 2-0 and player A is the winner.

a. Write a function `displayGameScore(gameScore)` that has a list in the above format as parameter and displays a summary game score. The result should be displayed in the following format:

```
Player A vs B
Game 1 21-11
Game 2 19-21
Game 3 11-21
Overall 1-2
Winner is player B
```

Test the function using any of the above lists.

b. Write a function `getPlayerNames()` that prompts for the names of 2 players and returns a game score list. The score list returned should be in the following structure:

```
[ [ ‘player 1 name’, ‘player 2 name’] ]
```

Since there are no game scores yet, the list consists of only the player names.

c. Write a function `inputGameScores(scoreList)` that has the score list as parameter and prompts for a game score. For example,

```
Game 1 score A vs B: 21-10
Game 2 score A vs B: 21-11
Game 3 score A vs B: <enter> key to represent end of input
```

(Assuming the players’ names are A and B)

The scores are entered with a dash in between. Add each game score in score list. Test out the function.

d. Write a main function to test out all 3 functions.

3. (Dictionary) Given the following dictionary structure that tracks the class sizes of tutorial groups in ICT133:

```
tutGp = {'T01':28, 'T02':15, 'T03':28, 'T04':25, 'T05':29, 'T06':22 }
```

The tutorial group name is the key and class size is the value. Write separate code for the following:

a. Print a tutorial group summary report. Each line has the tutorial group followed by the class size as follows:

```
TG Size
T01 28
T02 15
T03 28
T04 25
T05 29
T06 22
Total number of students 147
```

b. Add/update a tutorial group. Prompt for a tutorial group. If the tutorial group exists, display the existing size and prompt for a size to add/subtract. A negative value indicates that the class size should be subtracted and a positive number adds to the class size. Class size must not go below 20 and cannot be more than 30.

   If the tutorial group does not exist, display a message 'New tutorial group!' and prompt for the class size to assign to the group and add this tutorial group to the dictionary. Example:

   <u>Run 1:</u>
   ```
   Enter tutorial group: T07
   New Tutorial Group.
   Enter class size: 25
   Tutorial Group T07 added with class size 25
   ```

   <u>Run 2:</u>
   ```
   Enter tutorial group: T01
   Tutorial group exists. Class size is 28
   Enter size to add/subtract: 1
   Class size for T01 adjusted to 29
   ```

c. Add 3 students to every tutorial group, but for those whose sizes exceed 30, cap the size at 30. Display the resultant tutorial group size after each addition. Example output:
   ```
   T01 adjusted to max 30
   T02 adjusted to 18
   T03 adjusted to max 30
   T04 adjusted to 28
   T05 adjusted to max 30
   T06 adjusted to 25
   ```

4. (Dictionary) This program makes use of a dictionary structure to track currency rates. The rates are all with respect to 1 SGD. Write the program in parts as follows:

   a. Create an initial currency dictionary called curss as follows:
   ```
   currs = {'USD': 0.73, 'RMB':5.01, 'HKD':5.73 }
   ```

   b. Create a menu as follows:
   ```
   Menu
   1.  Add Currency
   2.  Adjust Currency
   3.  Remove Currency
   4.  Display Currency rates
   0.  Quit
   Enter option:
   ```
   For each option, call one of the functions described below.

   c. Function `addCurrency(currs).` Pass the currs dictionary to the function. The function should prompt user to input a currency and rate, e.g.
   ```
         Enter currency: MYR
         Enter rate: 2.90
   ```

The currency and rate are added to the curRs dictionary as a key value pair. If the currency already exists in the dictionary, print ‘Currency already exists!’, otherwise, proceed to include the currency in the dictionary.

d. Function `adjustCurrency(currs)`.
The function should prompt user to input a currency, e.g.
```
Enter currency: **HKD**
Rate is 5.73
Enter new rate: **5.77**
HKD adjusted to 5.77
```

The program checks that the currency exists before prompting for new rate. A message ‘Currency not found!’ should be displayed if the currency is not found.

e. Function `removeCurrency(currs)`. The function prompts for currency and if found, remove the currency from the dictionary.

f. Function `displayCurrencyRates(currs)`. The function displays the currencies in the following format:
```
Currency    Rate
USD         0.73
RMB         5.01
HKD         5.73
```

5. (Dictionary, list) Write a program to manage a collection of student names and their course marks. Course mark consists of 2 components – course work, and exam. Both are of equal weightage. Implement the program as described:

a. Assume that student names have been read from a file. Use the following initial dictionary structure:
```
marks = { 'John':[0,0], 'Jane':[0,0], 'Peter':[0,0], 'Joe':[0,0] }
```

Note that for each dictionary entry, name is the key and a list representing the coursework and exam marks is the value.

b. Allow user to repeatedly select an option from this menu:
```
Menu
1. Add marks
2. Update marks
3. Remove student
4. Display marks
0. Exit
```

c. Add marks option.
Prompt user for a name, coursework and exam. If the name already exists, display a message, otherwise add an entry to the dictionary with the name as key and coursework and exam score as values. An example run of the option is as follows:
```
Enter name: **John**
Coursework: **60**
Exam: **0**
Added!
```

d. Update marks option
   Similar to add, prompt user for a name, coursework and exam. However, the name must already exist before user is prompted for coursework and exam. An example run of the option is as follows:

```
Enter name: John
(John found. Marks displayed)
Coursework: 60
Exam: 70
Update C or E: C
Enter Coursework: 65
Updated!
```

e. Remove student option
   Prompt for a name and remove the entry if the name is in the dictionary.

f. Display mark option
   List all the names and scores of students in the following format:

```
Name  Cw    Ex    Overall Grade
John  60    70    65.0    P
Jane  50    40    45.0    F
```