CMPT 175- Lab Exercises

Week 2:

Q1 Write a Python program that repeatedly asks the user to input coin values (1 or 5 or 10 or 25) until the total amount matches a target value. The **target value** is a randomly generated integer between 1 and 99 inclusively. For example, for a target value of 31, coin values entered by the user should be **25, 5,** and **1** because 31 = 25 + 5 + 1.

How to handle input and input validation: When a game session starts, the player is asked to enter a valid coin value (1 or 5 or 10 o 25). If the player enters a coin value that is not valid (i.e. the coin value is not 1 or 5 or 10 or 25) the program prompts the player to enter a valid value. The program continues to prompt the player to enter a valid value until the player enters a valid value. If the player presses <Enter> key without entering a value the game session ends and the outcome of the game is reported.

A game session in the game ends with success or failure. The player attempt is successful if the player is able to enter valid coin values that add up to the total coin value. The player attempt fails if the player enters coin values that add up to **more than or less** than the total coin value.

In event of failure an appropriate message indicates by how many cents did the player exceeded the target amount or by how many cents was the player short of the target amount. Once the outcome (success or failure) has been reported the player is prompted to enter the letter 'y' or the letter 'n' to indicate whether they wish to play another game session. If the player enters 'y' a new game session with a new random coin value is started. If the player enters any value other than 'y' the program ends.

A complete sample run of the program is shown below

```
Game Session Starts
Enter coins values as 1-penny, 5-nickel, 10-dime, and 25-quarter.
Enter coins that add up to 18 cents, one per line.
Enter a valid coin value > 2
Invalid entry - Try again!
Enter a valid coin value > 10
Enter a valid coin value > hello
Invalid entry - Try again!
```

```
Enter a valid coin value >
Session Ends!
Game Session Ends
Here is the outcome :
Failure - you only entered 10 cents
You are short of 8 cents
Play another game session (y/n)?y
Game Session Starts
Enter coins values as 1-penny, 5-nickel, 10-dime, and 25-quarter.
Enter coins that add up to 78 cents, one per line.
Enter a valid coin value > 25
Enter a valid coin value > 25
Enter a valid coin value > 25
Enter a valid coin value > 10
Game Session Ends
Here is the outcome :
Failure - you entered 85 cents
The amount exceeds 78 cents by 7 cents
Play another game session (y/n)?y
Game Session Starts
Enter coins values as 1-penny, 5-nickel, 10-dime, and 25-quarter.
Enter coins that add up to 86 cents, one per line.
Enter a valid coin value >
Session Ends!
Game Session Ends
Here is the outcome :
Failure - you only entered 0 cents
You are short of 86 cents
Play another game session (y/n)?y
Game Session Starts
Enter coins values as 1-penny, 5-nickel, 10-dime, and 25-quarter.
Enter coins that add up to 81 cents, one per line.
Enter a valid coin value > 25
Enter a valid coin value > 25
Enter a valid coin value > 25
Enter a valid coin value > 5
Enter a valid coin value > 1
Game Session Ends
Here is the outcome :
Success!
Play another game session (y/n)?n
Thanks for playing ... goodbye
```

Q2 A tuple is sequence of comma separated values inside parenthesis. For instance (2,3) is a two-tuple. Write a function called unzip that takes in a list of two-tuples in its parameters and returns a tuple of two lists.

call to unzip([(1,4),(2,5),(3,6)]) would return ([1,2,3],[4,5,6]) .
a call to unzip([(1,2)]) would return ([1],[2])
a call to unzip([(‘A’,’B’),(‘X’,’Y’)]) would return ([‘A’,’X’],[‘B’,’Y’])
a call to unzip([]) would return ([],[])

Test your function by calling the function from inside the main function.