***BE 1600***

***Introduction to***

***Programming and Computation***

***Python***

**Assignment 06**

40 points

**Due 11/13/2023 (11:45 A.M.)**

Assignment Objectives:

* To understand Python lists
* To use Python lists as a means of storing data
* To demonstrate the use of list methods and operators
* To use tuples as immutable lists
* To give examples of representing data using two-dimensional lists.
* To use nested loops

*Solution for this assignment will not be posted on Canvas; however, the solution of any of the assignment problems can be discussed in the class upon request of a student.*

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it. State your answer clearly.
2. For programming questions, include only the source file of each programming problem.
3. Submit your files to the Canvas. You must submit your files on time; otherwise, you will receive zero.
4. Use “Add Another File” feature on Canvas to upload each additional file; do not upload the files as a compressed folder.
5. You can upload your files as many times as you like. Only the last attempt counts because the last files you uploaded are the only files your instructor will see.
6. There will be several modules on the Canvas. You need to upload your files using the correct module on the Canvas.
7. Name each file: *Assignment (assignment number)* for the word file [e.g. Assignment 02] and *Assignment (assignment number) \_ (Question number)* for each programming question [e.g. Assignment 02\_Q03].
8. To upload your file(s):

* In Course Navigation, click the ASSIGNMENTS module.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

*It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.*

***Make sure you review the Cheating & Plagiarism policy on Canvas.***

Write a program for questions Q.1. to Q.9. Save the files as text files. Submit 9 .txt files to Canvas by the due date.

**Question 01 (5 points)**

Write a **function** that takes a list of integers and counts and prints the occurrences of each. Write a main **function** that reads some integers between 1 and 100 and call the above function. *Do not use dictionary.*

Here is a sample run:

**Enter the integers between 1 and 100: 2 5 6 5 4 3 23 43 2**

**2 occurs 2 times**

**3 occurs 1 time**

**4 occurs 1 time**

**5 occurs 2 times**

**6 occurs 1 time**

**23 occurs 1 time**

**43 occurs 1 time**

**Question 02 (4 points)**

You can shuffle a list using *random.shuffle(lst).*

Write your own **function** without using *random.shuffle(lst)*to shuffle a list and return the list. Use the following function header:

defshuffle(lst):

Write a main **function** that prompts the user to enter a list of numbers, invokes the function to shuffle the numbers, and displays the numbers.

Here are sample runs:

**Enter numbers: 1 2 3 4 5 6 7 8 9**

**[4, 9, 5, 6, 1, 8, 2, 3, 7]**

**Enter numbers: 2 4 6 8 10 12 14 16 18**

**[16, 10, 2, 8, 18, 14, 12, 4, 6]**

**Question 03 (4 points)**

Write a **function** that returns a new list by eliminating the duplicate values in the list. Use the following function header:

*def**eliminateDuplicates(lst):*

Write a main **function** that reads in a list of integers, invokes the function, and displays the result. Do not use sets methods.

Here is a sample run:

**Enter numbers: 1 2 3 2 1 6 3 4 5 2**

**The distinct numbers are: [1, 2, 3, 6, 4, 5]**

**Question 04 (5 points)**

Write the following **function** that returns true if the list is already sorted in increasing order:

defisSorted(lst):

Write a main **function** that prompts the user to enter a list and displays whether the list is sorted or not.

Here are sample runs:

**Enter numbers: 1 1 3 4 4 5 7 9 10 30 11**

**The list is not sorted**

**Enter numbers: 1 1 3 4 4 5 7 9 10 30**

**The list is already sorted**

**Question 05 (4 points)**

Write a **function** that takes a tuple as an argument and returns the tuple sorted. Write a main **function** that prompts the user to enter 10 integers on the same line, creates a tuple from the 10 integers, and prints the tuple before and after the above function is called.

Hint: use a list to store the 10 integers and convert the list to a tuple.

Here is a sample run:

**Enter 10 integers on the same line 4 2 8 1 3 6 9 7 5**

**Before sorting (4, 2, 8, 1, 3, 6, 9, 7, 5)**

**after sorting (1, 2, 3, 4, 5, 6, 7, 8, 9)**

**Question 06 (5 points)**

Write a **function** that returns the sum of all the elements in a specified column in a matrix using the following header:

defsumColumn(m, columnIndex):

Write a main **function** that prompts the user to enter a 3-by-4 matrix of numbers, calls the above function, and displays the sum of each column.

Here are sample runs:

**Enter a 3-by-4 matrix row 0: 1.5 2 3 4**

**Enter a 3-by-4 matrix row 1: 5.5 6 7 8**

**Enter a 3-by-4 matrix row 2: 9.5 1 3 1**

**Sum of the elements at column 0 is 16.5**

**Sum of the elements at column 1 is 9.0**

**Sum of the elements at column 2 is 13.0**

**Sum of the elements at column 3 is 13.0**

**Question 07 (5 points)**

An *n x n* matrix is called a *positive Markov matrix* if each element is positive and the sum of the elements in each column is 1. Write the following **function** to check whether a matrix is a Markov matrix:

def isMarkovMatrix(m):

Write a main **function** that prompts the user to enter a matrix of numbers and tests whether it is a Markov matrix.

Here are some sample runs:

**Enter a 3 by 3 matrix row by row:**

**0.15 0.875 0.375**

**0.55 0.005 0.225**

**0.30 0.12 0.4**

**It is a Markov matrix**

**Enter a 3 by 3 matrix row by row:**

**0.95 -0.875 0.375**

**0.65 0.005 0.225**

**0.30 0.22 -0.4**

**It is not a Markov matrix**

**Question 08 (4 points)**

Write a main **function** that prompt the user for the expenses for last month in the following categories: Rent, Gas, Food, Clothing, Car Payment, and Misc. The function stores the expenses in a list and uses matplotlib to plot a pie chart showing how the user spend the money.

Here is a sample run:

**Enter the expenses for last month in the following categories:**

**Rent: 1250**

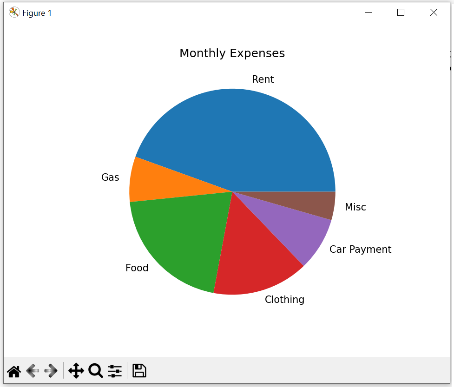
**Gas: 200**

**Food: 575**

**Clothing: 425**

**Car Payment: 235**

**Misc: 125**



**Question 09 (4 points)**

Write a main **function** that uses matplotlib to display a line graph of average monthly high temperature for particular city in a particular year. The function should first ask for a year, then asks for the temperature for each month of that year. Be sure to display meaningful labels along X and Y axes, as well as the tick marks and grid.

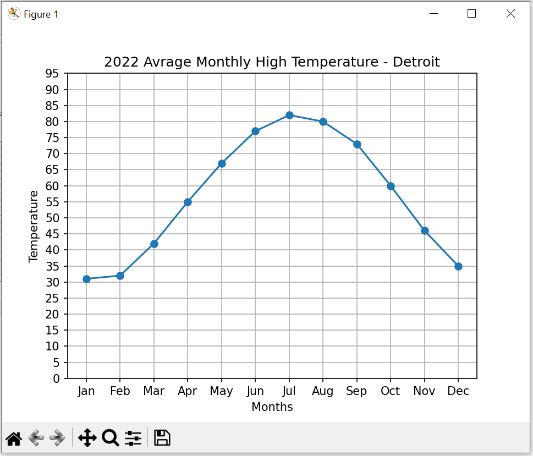
Here is a sample run:

**Enter city: Detroit**

**Enter year: 2022**

**Enter average high temperature for each month of the year:**

**31 32 42 55 67 77 82 80 73 60 46 35**

****