Section 14

PDS Lab **Lab – 5** **05.01.2022**

Instructions:

- Give sufficient comment against each statement in your program.
- You should save each program with the file name (e.g., Lab5_1.c for the program of Problem 1 in this assignment).
- There is a partial credit even if your program does not run successfully for all the test cases as mentioned.
- There are FIVE problems and you have to solve in 150 minutes. A tentative time against each problem is given and can be considered for your guidance.
- You should upload each program (only .c file) against the problem. There is no need to submit any .zip file at the end of your lab.

1. Write a program in C for the following.
   (a) Define an array of integers. Your array should store a sufficiently large number of values.
   (b) Initialize the array reading only integer values from the keyboard. If a user enters a character/floating point number your program should report a warning message and skips that entry. Your program stops when user types 0 (zero).
   (c) Find the average of all **unique** numbers in the array.
   (d) Find the standard deviation of the whole list.
   Hint: Standard deviation of n numbers $x_1, x_2, \ldots, x_n$ is

$$STD = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

   where, $\bar{x}$ is the average value of the numbers.

Run your program with the following test cases:

```
Test Case# 1:
Input: [5, 10, 15, 20, 0]
Output: Average of unique numbers =12.50
        Standard deviation of whole list=2.80

Test Case# 2:
Input- [5, 10, 15, 20, 5, 0]
Output: Average of unique numbers=12.50
        Standard Deviation of whole list=2.61

Test Case# 3:
Input - [1, 2.2, 3, 4, 5, -6, 7, a, 0]
Output:
Warning message: "Float value entered and skipped."
                 "Character value entered and skipped."
Output: Average of unique numbers = 2.33
        Standard deviation of whole list = 1.69
```

[Time: 30 minutes] [4× 2.5+3×5]

2. An n-dimensional vector can be considered as (x_1, x_2, …., x_n). Do the following tasks in a single program but one after another:

   (a) Read a vector, say v from the keyboard. After reading a vector, your program should print the magnitude $|v|$ of the vector v.

   Hint: $|v| = \sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}$

   (b) Read two vectors v_1 and v_2. For the two vectors, calculate the dot product of two vectors.

   Hint: $v_1.v_2 = x_{11}.x_{21} + x_{12}.x_{22} + \cdots + x_{1n}.x_{2n}$

   (c) Two vectors v_1 and v_2 as read in the last task, find the angle between two vectors v_1 and v_2.

   Hint: $cos\theta = \frac{v_1.v_2}{|v_1|\,|v_2|}$

```
Test Case# 1:
Input: v1: [1, 0 , 0, 1, 0]   v2: [0, 1, 1, 0, 1]
Output: angle= 90 degree

Test Case# 2:
Input Vectors: v1=[1, 1, 1, 1]    v2=[5, 5, 5, 5]
Output: 0 degree

Test Case# 3:
Input Vectors: v1=[1, 0, 0] v2=[1.41, 1.41, 0]
Output: 45 degree

Test Case# 4: V1=[1, 0, 0, 0 ]    V2=[1, 0]
Output : "Error: Length of the two vectors must be equal"
```

[Time: 30 minutes] [(3×5)+4× 2.5]

3. Read an array of *n* number of integer values ($n > 0$) from the keyboard and store them in an array, say *list*. Then do the following:
   (a) Number of negative, positive and zero values in the array.
   (b) Check if the array is in sorted order or not. Please don't sort the numbers. You have to test without sorting.
   (c) If the array is in sorted order, then check if it is in ascending or descending order.

```
Test Case# 1:
Input: list: [1]
Output: Number negative values = 0
        Number of positive values = 1
        Number of zeros  = 0
        List is sorted!

Test Case# 2:
Input: list: [-1  2  -3  0  4  -5  6  9]
```

```
Output: Number of negative values = 3
        Number of positive values = 4
        Number of zeros  = 1
        List is not sorted.
Test Case# 3:
Input: list: [9  6  5  3  0  -1  -2  -3]
Output: Number of negative values = 3
        Number of positive values = 4
        Number of zeros  = 1
        List is sorted in descending order.

Test Case# 4:
Input: list: [-3  -2  -1  0  1  2  3  4  5  6]
Output: Number of negative values = 3
        Number of positive values = 6
        Number of zeros  = 1
        List is sorted in ascending order.
```

[Time: 30 minutes] [(3×5)+4× 2.5]

4. Read an array of *n* number of integer values ($n > 0$) from the keyboard and store them in an array, say *list*. Then do the following:
   (a) Read any integer, say *what* from the keyboard. Check if what is present in *list* or not.
   (b) If *what* is present in the list, then count how many comparison your program took to find it what is its position in the list.
   (c) If *what* is not present in the list, then place it as the first element in *list* and then print *list*.

```
Test Case# 1:
Input: list: [85  55  65  75  95]
Output: what = 65
        What is present in the list
        What is present at location 3
        Number of comparisons  = 3


Test Case# 2:
Input: list: [85  55  65  75  95]
Output: what = 15
        what is NOT present in the list
        Number of comparisons  = 5
        list with what  is [15  85  55  65  75  95]
```

[Time: 30 minutes] [(3×5)+2×5]

5. Read an array of *n* number of floating point values ($n > 0$) from the keyboard and store them in an array, say *list*. Then do the following:
   (a) Sort the values. Print the array both before and after sorting. You should not use any additional array in your program.
   (b) Calculate the average value of the values in the sorted array. Let the average value be *average*.
   (c) If *average* is not present in the array, then place it in the array so that all values in sorted order. Print the array after adding average.

```
Test Case# 1:
Input: list: [8.5  5.5  6.5  7.5  9.5]
Output: list before sort = [8.5  5.5  6.5  7.5  9.5]
        list after sort =  [5.5  6.5  7.5  8.5  9.5]
        average =  7.5
        average is present in the list


Test Case# 2:
Input: list: [3.0  2.0  1.0  6.0  5.0  4.0]
Output: list before sort = [3.0  2.0  1.0  6.0  5.0  4.0]
        list after sort =  [1.0  2.0  3.0  4.0  5.0  6.0]
        average =  3.5
        average is NOT present in the list
     list with average is: [1.0 2.0 3.0 3.5 4.0 5.0  6.0]
```

[Time: 30 minutes] [(3×5)+2×5]

---*---