Week 3: DAY 2 -PROBLEM SOLVING WITH ARRAYS

Part-A: Initialization (Ending with -1)

1. Declaring and visualizing an array, an Initializing and array. In many ways
    a. Initializing with 0's
    b. Initializing with i
    c. Initializing with 2i
    d. Initializing with k*i + 1
    e. Initializing with Prime numbers.
    f. Randomly Initializing

Part-B: Searching Problems

1. Find the first index of a specific element in an array. (Passing to functions)

    Sample Input:

    Enter the Array. Enter -1 to exit: 2 589 512 -4 -6 1 1 4 5 6 -1

    Enter the Element to Search: 1

    Sample Output:

    The index of 1 is: 5
2. Find the last index of a specific element in an array.

    Sample Input:

    Enter the Array. Enter -1 to exit: 1 589 512 -4 -6 1 4 56 -1

    Enter the Element to Search: 1

    Sample Output:

    The index of 1 is: 5
3. Find the index of the k^{th} existence of an element in an array.

    Sample Input:

    Enter the Array. Enter -1 to exit: 1 589 512 -4 -6 1 4 56 -1

    Enter the Element to Search: 1

    Enter the value of k for k^{th} index: 1

    Sample Output:

    The k^{th} index of 1 is: 5
4. Find the index of the k^{th} last existence of an element in an array.

    Sample Input:

    Enter the Array. Enter -1 to exit: 1 589 512 -4 -6 1 4 56 -1

    Enter the Element to Search: 1

    Enter the value of k for k^{th} last existence index: 1

    Sample Output:

    The index of 1 is: 0

Part C: Frequency based Problems

1. Write a function which returns the frequency of a given element in an array

    Sample Input:

    Enter the Array. Enter -1 to exit: 4 3 9 1 51 4 4 7 7 3 7 9

    Enter the element: 4

    Sample Output:

    Frequency of 4 is 3
2. Write a function which counts the frequency of each element of an array [using previous function.]

    Sample Input:

    Enter the Array. Enter -1 to exit: 4 3 9 1 51 4 4 7 7 3 7 9

Sample Output:
Frequency of 4 is 3
Frequency of 3 is 2
Frequency of 1 is 1
Frequency of 51 is 1
Frequency of 7 is 3
Frequency of 9 is 2

3. Write a program which displays the prime frequency of each element within a given range. If the given range index is not within the array boundary (Starting Index of an array or Ending Index of an array), select the nearest boundary index.

Sample Input:
Enter the Array. Enter -1 to exit: 4 3 9 1 51 4 4 7 7 3 7 9
Enter Starting Range Index: 2
Enter Ending Range Index: 7
Sample Output:
Frequency of 2 is prime frequency which is 2
Note: Prime frequency of an element means the number of occurrences of that element should be a prime number.

Part D: Finding Unique and Distinct Element

Write a program that takes upto 20 integers (Capacity) …….. with -99.

1. Further your program should be able to identify the distinct elements and store it in an array named as DistinctArray and then using the print function it should display the DistinctArray. Distinct elements of an array are such that if an element appears more than once, then it should be printed once only.
2. Further your program should be able to identify the unique element and store it in an array named as UniqueArray and then using the print function it should display the UniqueArray. Unique elements of an array are the ones which occur only once in an array.
3. Now your program should make sure that the DistinctArray should be sorted in increasing order.

Sample Input:
20 11 12 20 16 15 12 16 8 12 -99
Sample Output:
Distinct Element in Sorted (Increasing order) are: 8 11 12 15 16 20
Unique Element in Sorted(Decreasing order) are: 15 11 8

Part E: Segregation based Problems

1. Segregate the odd and even values in an array.

Sample Input:
Enter the Array. Enter -1 to exit: 1 589 512 -4 -6 1 4 56 -1
Sample Output:
Segregated Array: 56 4 -6 -4 512 1 589 1

2. You have an array of zeros and ones, move all zeros to the left and ones to the right.

Sample Input:
Enter the Array. Enter -1 to exit: 1 0 0 1 0 1 0 0 0 -1
Sample Output:
Segregated Array: 0 0 0 0 0 0 1 1 1

3. You are given an array of integers and a value V, segregate all the smaller than V values to the left and bigger values to the right.

Sample Input:
Enter the Array. Enter -1 to exit: 1 589 512 -4 -6 1 4 56 -1
Enter the Segregation Value: 56

Sample Output:

Segregated Array: 1 -4 -6 1 4 56 512 589

4. You have to read three letters representing three fruits 'b' for a banana, 'm' for mangoes and 'a' for apples, now segregate the values in such a way that all b's are in the beginning and then m's mangoes and then a's apples.

   Sample Input:

   Enter the Array. Enter -1 to exit: b m m m a a b a m a m b b b a

   Sample Output:

   Segregated Array: b b b b b m m m m m a a a a a

   Bonus: In all of the above questions keep the order in which they were originally present.

5. You are given an array of Prime, Fibonacci and Non-PrimeFibonacci (the numbers neither prime nor fibonacci) numbers in random order. Segregate all Fibonacci on left side, Prime on right side and Non-PrimeFibonacci in the middle of the array. If a number is Prime and Fibonacci both, it's your choice to treat it as a Prime or Fibonacci number. Also find the MinimumSwapsCount - the number of swaps required to get the final array.

Sample Input: Size: 13

INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12
VALUE	2	5	4	17	34	11	10	8	18	24	23	89	1

Sample Output

INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12
VALUE	1	5	89	8	34	4	10	24	18	17	23	11	2

Minimum Swaps Count: 5

Fibonacci elements are {1, 5, 89, 8, 34}, NonPrimeFibonacci elements are {4, 10, 24, 18} and Prime elements are {17, 23, 11, 2}

Part F: Merging Arrays

1. Write a function which takes two arrays, **arr1, arr2** and their sizes, **s1, s2** as parameters. Your function should return a third array **arr3** of size **s3 = s1 + s2** which contains all elements of both arrays.

   Sample Input:

   Enter the size of Array1: 5

   Enter the Elements of Array1: 2 4 3 7 11

   Enter the size of Array2: 7

   Enter the Elements of Array2: 12 4 33 27 0

   Sample Output:

   Merged Array3 of Array1 and Array2: 2 4 3 7 11 12 4 33 27 0

Part G: Sorting

- ❏ **Bubble Sort** ------- **Implementation did in Class-------**
- ❏ **Selection Sort** ------- **Algorithm Discussed in Class ----- Try to Implement it**
- ❏ **Insertion Sort**

Sort an array Arr[] of size S using Insertion Sort. For this you are required to implement two functions:

1. **InsertInSortedArray(int Arr[], int i)** function which inserts an element in a sorted array. It takes two parameters, an array Arr[] and a i^{th} index - the index of new element which you wants to insert in sorted manner. This function compares the i^{th} index element with all previously sorted indices' elements (with $i\text{-}1^{th}$ till starting index 0). If the i^{th} index element is smaller than $i\text{-}1^{th}$ index element, a Swap (arr[i], array[i-1]) should happen and it repeatedly swaps with previous elements until the previous element is smaller or you reach at array's starting index - 0.

2. InsertionSort(int Arr[], int S) function sort all unsorted array's elements using InsertInSortedArray() function. It passes all array elements' indices one by one till size S. After each InsertInSort() function call, the array should be sorted in ascending order till i^{th} index.

- *First make InsertInSortedArray() and then from InsertionSort() function, call InsertInSortedArray() for each i^{th} index value one by one*

Sample Input : S= 6

INDEX	0	1	2	3	4	5
VALUE	52	12	3	14	17	10

After i = 0, InsertInSortedArray(Arr, 0)

INDEX	0	1	2	3	4	5
VALUE	52	12	3	14	17	10

After i = 1, InsertInSortedArray(Arr, 1)

INDEX	0	1	2	3	4	5
VALUE	12	52	3	14	17	10

After i = 2, InsertInSortedArray(Arr, 2)

INDEX	0	1	2	3	4	5
VALUE	3	12	52	14	17	10

After i = 3, InsertInSortedArray(Arr, 3)

INDEX	0	1	2	3	4	5
VALUE	3	12	14	52	17	10

After i = 4, InsertInSortedArray(Arr, 4)

INDEX	0	1	2	3	4	5
VALUE	3	12	14	17	52	10

After i = 5, InsertInSortedArray(Arr, 5)

INDEX	0	1	2	3	4	5
VALUE	3	10	12	14	17	52

Sample Output:

INDEX	0	1	2	3	4	5
VALUE	3	10	12	14	17	52

Part H: Binary Search

1. Recall the concept of binary search we did in the homework 1.
   i.e.Finding out the heaviest ball using a balance.
   Now try to implement what you have learned in this homework.

Part I: Adding an Element at ith Index

1. You are given an array and an index. You have to add an element at the given index of the given array and print the new array .

   Sample Input : -6 -4 0 1 4 56 589 index = 2 , element = -2
   Sample Output: -6 -4 -2 0 1 4 56 589

Part J: Sorting with Respect to their Frequencies

1. Write a program which sorts the given data with respect to their frequency.
   Hint: Do exactly the same process as you did for finding the unique first Us. Then make another array Freq holding frequencies, in which you should populate the frequency of each element in a unique array and save it as a Freq array. Now sort Us not on the basis of values but their corresponding frequency in Freq, the element

with highest frequency should appear first in the Us array. Now replace all the values inside the data array D, by replacing each Us value one by one, as many times as their frequency is there in the Freq array.

Sample input:
Input Array: 2 5 76 53 2 89 4 76 2 2 43 53 53 2 89 76 76 -99
Sample Output:
Us: 2 5 76 53 89 4 43
Freq: 5 1 4 3 2 1 1
Sorted Us: 2 76 53 89 5 4 43
New D Array: 2 2 2 2 2 76 76 76 76 53 53 53 89 89 5 4 43

Part K: Computing Leaders

1. Write a program that takes upto 20 integers (Capacity) as input from the user and prints the numbers in the same order as entered. If a user wants to enter less than 20 numbers, the user shall terminate it with -99.
   Further your program should be able to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a LEADER.
   Now instead of just printing, store the LEADERS in an array by making a proper function with an array of a large capacity and Size should be set with a number of elements written in the LEADERS array. After that print LEADERs using a print function (with array as parameter passed).

   Sample Input : 20 11 12 14 16 15 19 13 18 17 -99
   Sample Output : 20 19 18 17

Part L: Computing Median of an Array

1. You are given an array. Find the median of elements given in the array. For simplicity assume there are no duplicates.

   Sample Input : -6 -4 0 1 2 4 56 512 589
   Sample Output: Median = 2

Part M: Rotating an Array

1. You are given an array and a number N. Your task is to perform N left rotations to the array.

   Sample Input : Array -6 -4 0 1 4 56 589
   Number: 3
   Sample Output: 1 4 56 589 -6 -4 0

Part N: Removing an Element from ith Index

1. You are given an array and an index. You have to remove the element at that index from the array and print the new array .

   Sample Input : -6 -4 0 1 2 4 56 512 589 index = 3
   Sample Output: -6 -4 0 2 4 56 512 589
   Sample Input : Index = 6
   Sample Output: -6 -4 0 2 4 56 589

Part O: Finding Equilibrium Index

1. Write a program that keeps on taking input from the user until the user enters -99 (at maximum 100 values). Further your program should return an equilibrium index of an array. If no equilibrium index is found then your program should return -1.

- An index of an array is said to be an equilibrium index if the sum of the elements on its lower indexes is equal to the sum of elements at higher indexes.

Sample Input:

INDEX	0	1	2	3	4	5	6
VALUE	7	-1	-5	2	4	-3	0

Sample Output: 3 is an equilibrium index (A[0] + A[1] + A[2] = A[4] + A[5] + A[6] = 1)

Part P: Sorting with evens and odds ascending and descending

1. Write a program that keeps on taking input from the user until the user enters -1 (at maximum 100 values) and then sort the even index values, in increasing order and odd index values, in decreasing order

Sample Input

INDEX	0	1	2	3	4	5	6	7
VALUE	100	10	2	3	27	9	19	13

Sample Output

INDEX	0	1	2	3	4	5	6	7
VALUE	2	13	19	10	27	9	100	3

Bonus Challenge 1:

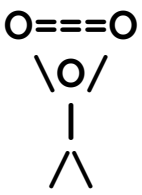
Given an array of integers, print and sort the array in such a way that the first element is first maximum and second element is first minimum and so on.

Sample Input: Size: 8

INDEX	0	1	2	3	4	5	6	7
VALUE	17	11	12	13	14	15	16	18

Sample Output

INDEX	0	1	2	3	4	5	6	7
VALUE	18	11	17	12	16	13	15	14



Happy Coding... :)

Good luck