Lab - Sorting variants and their applications

Part A: Sorting

- **Bubble Sort ------- 3 Implementation did in Class-------**
  - i. Using two loops
    - a. Using two nested loops
    - b. Using two nested loop with arithmetic series first small improvement.
  - ii. Using for and a while loop, with improvement of introducing a changeHappen variable.
  - iii. Using bubbling function
- **Selection Sort (Two implementations)**
  - i. Using two nested loop
  - ii. Using one loop and rangeMinIndex function
- **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: Final

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

Part B: **Search** or sorting arrays

1. Given a sorted array (can you modify the linear search such that it should not search beyond the value which we are looking for.
2. Recall the concept of binary search we did in homework 1 i.e.Finding out the heaviest ball using a balance. Now try to implement what you have learned in this homework.

3. Using Binary search, compute the square-root, a given number (to the nearest 6 decimal places).
4. Using Binary search, compute the third-root, a given number (to the nearest 6 decimal places).
5. Using Binary search, compute the k'th-root, a given number (to the nearest 6 decimal places).

Part C: 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

2. 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	14	13	12	11	15	16	18

Sample Output

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

3. Given an array of integers, sort the array in such a way that the even values are in ascending order and odd values are in descending order. Note: Even values will be compared with only even values and odds with odds.

Sample Input: Size: 8

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

Sample Output

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

Part D: Electronic Voting

Election happened in a country with 8 parties fighting inside the election. Make a program to check who won the general election. Read the data from Votes.txt file.

NOTE: Votes must be read from the file. The PVotes(Party votes count) Array can be handled using two different arrays one for IDs and one for Frequencies(vote count casted to each Party) both should be handled with maximum capacity and size variables. You may assume that the number of votes in a constituency couldn't exceed 1000.

Expected **OUt** ⇒ ⇒ ⇒ ⇒ ⇒ ⇒

Votes.txt

8
PPP PML PTI ANP JIP JUI MQM PAT
40
4 1 2 7 3 2 1 7 5 6 6 6 2 1 1 6 7 5 6 3 6 6 1 1 5 8 4 7 1 5 5 1 1 1 7 1 6 6 1 8

```
O===O
 \O/
  |
```

```
Party Name      Parties      Votes
PPP             1            12
PML             2            3
PTI             3            2
ANP             4            2
JIP             5            5
JUI             6            9
MQM             7            5
PAT             8            2

PPP ████████████
PML ███
PTI ██
ANP ██
JIP █████
JUI ████████
MQM █████
PAT ██

PPP party won the election.
```

/\

Happy Coding... :)

Good luck