LAB 13: Sorting-2

Statement Purpose:

Purpose of this Lab is to familiarize the students with the use of N-Squared sorting algorithms namely Selection Sort and Bubble Sort. Another aim is to teach the students how we can improve sorting time using Merge Sort algorithm. The students are given small tasks related to aforementioned sorting techniques under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

Activity Outcomes:

The students will learn how various sorting techniques e.g. Selection Sort, Bubble Sort, Merge Sort and Quick Sort work on given data set. This will help them understand the difference among these sorting techniques.

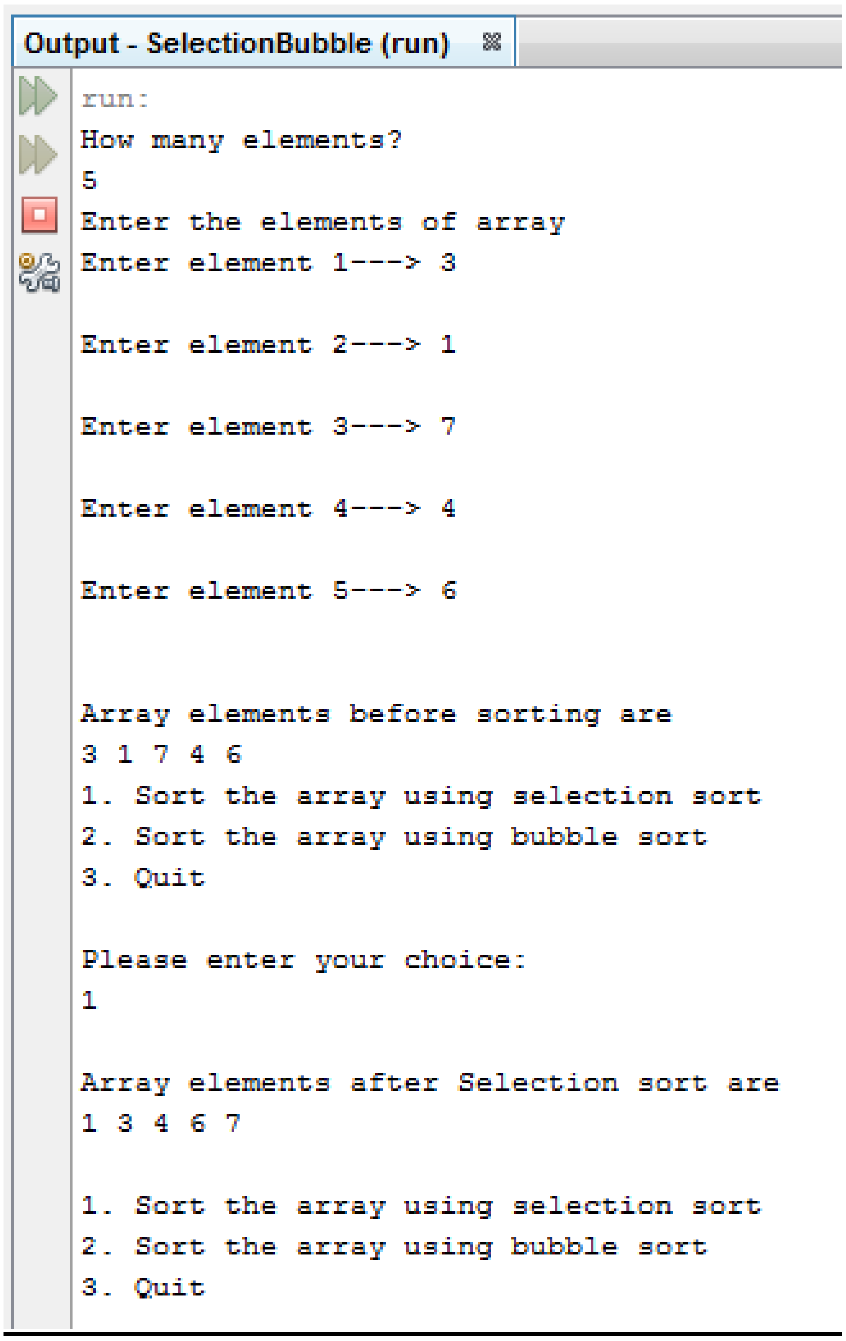
Lab Activities: (40 Minutes)

1. Following is an algorithm of Selection Sort. Your task is to write a complete java program which takes 'n' elements from user at run time and stores in an array of length 'n'. Your program should use the following Selection Sort algorithm and then display the elements of array in **ascending order**.

Selection Sort Algorithm:

```
Let the elements be stored in an array "a"
Let n = a.length

for ( i = 0 ; i< n ; i ++ )
{
        Find the array element with the min. value among a[i], ...,
a[n-1];
        Swap this element with a[i];
}
```

Sample Run:



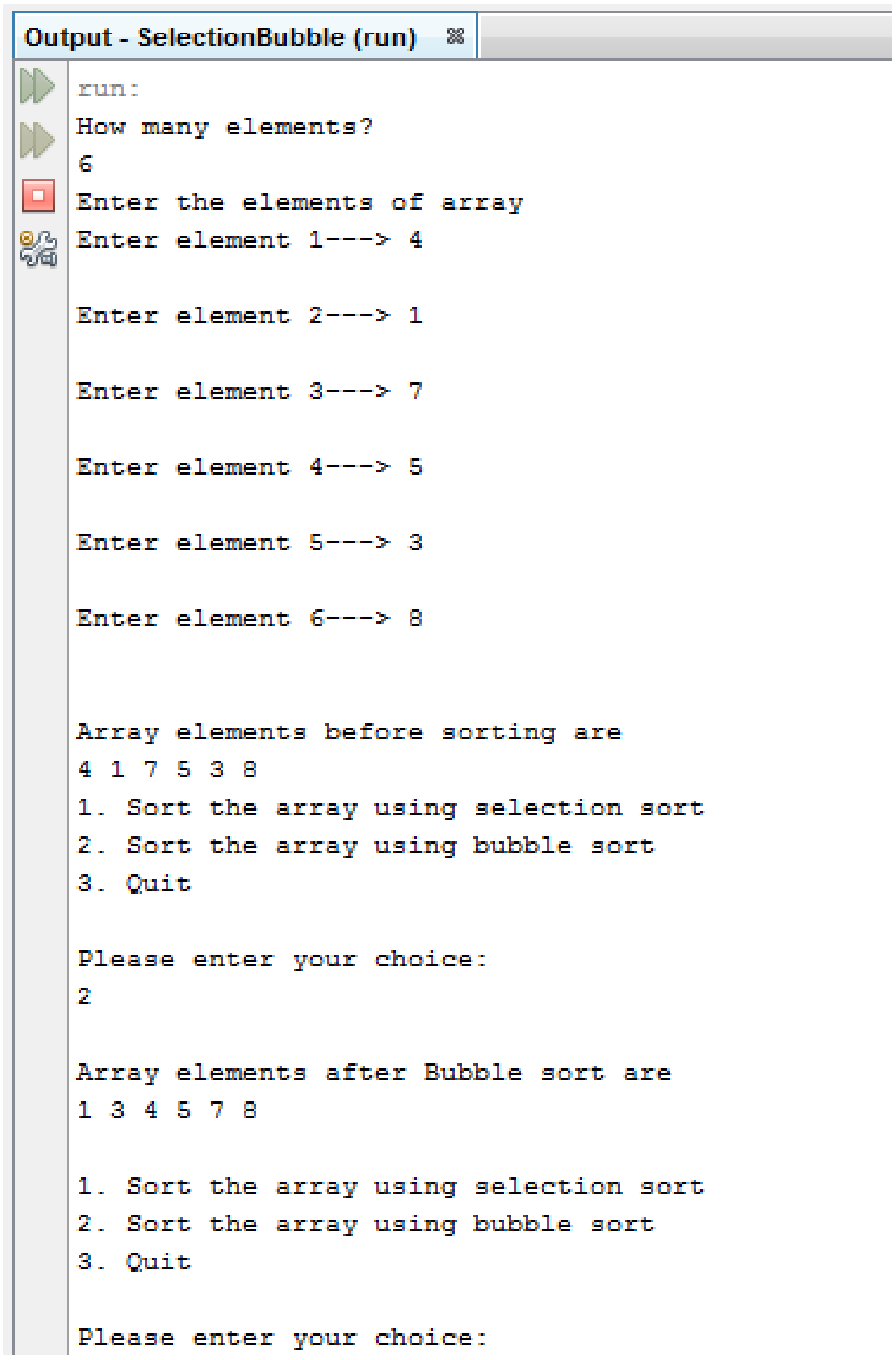
```
run:
How many elements?
5
Enter the elements of array
Enter element 1---> 3

Enter element 2---> 1

Enter element 3---> 7

Enter element 4---> 4

Enter element 5---> 6


Array elements before sorting are
3 1 7 4 6
1. Sort the array using selection sort
2. Sort the array using bubble sort
3. Quit

Please enter your choice:
1

Array elements after Selection sort are
1 3 4 6 7

1. Sort the array using selection sort
2. Sort the array using bubble sort
3. Quit
```

2. Following is an algorithm of Bubble Sort. Your task is to write a complete java program which takes 'n' elements from user at run time and stores in an array of length 'n'. Your program should use the following Bubble Sort algorithm and then display the elements of array in **ascending order**.

Bubble Sort Algorithm:

```
procedurebubbleSort( A : list of sortable items )
      n = length(A)
      boolean swapped = true;
      while (swapped == true) do
            swapped = false
            fori = 1 to n-1 inclusive do
      /* if this pair is out of order */
            if A[i-1] > A[i] then
      /* swap them and remember something changed */
                  swap( A[i-1], A[i] )
                  swapped = true
            end if
            end for
      end while
end procedure
```

Sample Run:


Output - SelectionBubble (run)

```
run:
How many elements?
6
Enter the elements of array
Enter element 1---> 4

Enter element 2---> 1

Enter element 3---> 7

Enter element 4---> 5

Enter element 5---> 3

Enter element 6---> 8


Array elements before sorting are
4 1 7 5 3 8
1. Sort the array using selection sort
2. Sort the array using bubble sort
3. Quit

Please enter your choice:
2

Array elements after Bubble sort are
1 3 4 5 7 8

1. Sort the array using selection sort
2. Sort the array using bubble sort
3. Quit

Please enter your choice:
```

Lab Exercises: (40 Minutes):

1) Show the contents of the array below being sorted using selection Sort at the end of each loop iteration. The array should be sorted in **descending order**.

Initial	7	5	1	4	2	6	8	3
1st Iteration								
2nd Iteration								
3rd Iteration								
4th Iteration								
5th Iteration								
6th Iteration								
7th Iteration								
Sorted	8	7	6	5	4	3	2	1

2) Show the contents of the array below being sorted using Bubble Sort at the end of each loop iteration. The array should be sorted in **descending order**.

Initial	7	5	1	4	2	6	8	3
1st Iteration								
2nd Iteration								
3rd Iteration								
4th Iteration								
5th Iteration								
6th Iteration								
7th Iteration								
Sorted	8	7	6	5	4	3	2	1

3) Show the contents of the array below after each merge occurs in the process of Merge-Sorting the array below. The array should be sorted in **descending order**.

Initial	10	5	25	50	20	35	40	45
1st Iteration								
2ndIteration								
3rdIteration								
4thIteration								
5thIteration								
6thIteration								
7thIteration								
Sorted	50	45	40	35	25	20	10	5

4) Show the result of running Partition on the array below using the leftmost element as the pivot element. Show what the array looks like after each swap. The array should be sorted in **descending order**.

Initial	5	2	1	4	7	8	3	6
After partition								