Lab 16: Learn to sort array using elementary sorting algorithms

Sorting: Sorting is the process of arranging the data in a particular order.

Time complexity of Bubble sort, Selection sort and Insertion sort is $\mathbf{n^2}$

Selection Sort Algorithm:

	8	5	10	3	1

Step 1:

	1	5	10	3	8

Step 2:

	1	3	10	5	8

Step 3:

	1	3	5	10	8

Step 4:

	1	3	5	8	10

Selection Sort Code:

```cpp
#include<iostream>
using namespace std;

void selectionSort(int arr[], int n){
    for(int i=0; i<n-1; i++){
        int smallestIdx = i; //unsorted part starting
        for(int j=i+1; j<n; j++){
            if(arr[j] < arr[smallestIdx]){
                smallestIdx = j;
            }
        }
        swap(arr[i], arr[smallestIdx]);
    }
}

void printArray(int arr[], int n){
    for(int i=0; i<n; i++){
        cout<<arr[i]<<"  ";
    }
}

int main (){
    int n = 5; //Array size
    int arr[]={8,5,10,3,1}; //unsorted array
    selectionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

```
1  3  5  8  10
--------------------------------
Process exited after 0.557 seconds with return value 0
Press any key to continue . . .
```

Insertion Sort Algorithm:

10	4	8	16	11

Step 1:

4	10	8	16	11

Step 2:

4	8	10	16	11

Step 3:

4	8	10	16	11

Step 4:

4	8	10	11	16

Insertion Sort Code:

```cpp
#include<iostream>
using namespace std;

void insertionSort(int arr[], int n){
    for(int i=1; i<n; i++){
        int curr = arr[i];  //Current element
        int prev = i-1;     //Previous
        while(prev >= 0 && arr[prev] > curr){
            arr[prev+1] = arr[prev]; //Shifting previous larger element to the right
            prev--;
        }
        arr[prev+1] = curr; //Placing the current element at its correct position
    }
}

void printArray(int arr[], int n){
    for(int i=0; i<n; i++){
        cout<<arr[i]<<"  ";
    }
}

int main (){
    int n = 5;
    int arr[]={10,4,8,16,11};
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

```
4  8  10  11  16
--------------------------------
Process exited after 0.4822 seconds with return value 0
Press any key to continue . . .
```

Bubble Sort Algorithm:

In bubble sort we compare adjacent elements.

1st iteration:

Step 1:

6	1	3	2	9

Step 2:

1	6	3	2	9

Step 3:

1	3	6	2	9

Step 4:

1	3	2	6	9

Step 5:

1	3	2	6	9

Note: After 1st iteration we can see that 1st largest element is moved to Last of the array.

2nd iteration:

Step 1:

1	3	2	6	9

Step 2:

1	3	2	6	9

Step 3:

1	2	3	6	9

Step 4:

1	2	3	6	9

Note: Our loop will execute 4 times means iterations will be 4 as total elements are n = 5 so number of iterations will be n-1 = 4. But in above condition our array is sorted within 2 iterations it doesn't means that loop exits after 2 iterations, loop will continue to execute upto 4 iterations. But in 4th iteration our number of steps will be reduced to 2.

Bubble Sort Code:

```cpp
#include<iostream>
using namespace std;

void bubbleSort(int arr[], int n){
    for(int i=0; i<n-1; i++){
        bool isSwap = false;
        for(int j=0; j<n-i-1; j++){
            if(arr[j] > arr[j+1]){      //Comparing adjusent elements
                swap(arr[j], arr[j+1]); //Pushing large element to the last
                isSwap = true;
            }
        }
        //To avoid extra iterations and make our code optimum
        if(!isSwap){//array is already sorted
            return;
        }
    }
}
void printArray(int arr[], int n){
    for(int i=0; i<n; i++){
        cout<<arr[i]<<"  ";
    }
}
int main (){
    int n = 5; //Array size
    int arr[]={6,1,3,2,9}; //unsorted array
    bubbleSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

```
1  2  3  6  9
--------------------------------
Process exited after 0.4733 seconds with return value 0
Press any key to continue . . .
```