SECJ2013 2526-1 LAB 2

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

void bubbleSort(int data[], int listSize)
{
    // implement the bubble sort algorithm


} // end bubble sort

void iBubbleSort(int data[], int listSize)
{
    // implement the improved bubble sort algorithm


} // end improved bubble sort

void selectionSort(int data[], int n)
{
    // implement the selection sort algorithm


} // end selection sort

void swap(int& x, int& y)
{

} // end swap

void insertionSort(int data[])
{
    // implement the insertion sort algorithm


} // end insertion sort

int main(){
    int dataArrayA[25] = { };
    int dataArrayB[25] = { };

    // Implement your code to call bubble sort, improved bubble sort,
    // selection sort, and insertion sort functions seperately.


}
```

1. Complete program above to sort 25 integers below into ascending order using simple sort algorithms: Bubble Sort, Insertion Sort and Selection Sort. For each sorting technique learned in class, modify the algorithms so that the programs are able to count and print the number of passes, the number of data comparisons and the number of data swapping that take place in the sorting process.

a. Run the programs on the following list of integer values. You can initialize the data in the array declarations.

int dataArrayA[25] = { .... };

int dataArrayB[25] = { .... };

100	50	88	30	60	45	25	12	10	5	98	15	65	55	45	70	20	90	66	22	120	48	35	85	5

dataArrayA

5	8	30	25	35	40	42	50	55	22	24	66	70	75	78	80	85	90	100	118	98	120	122	121	121

dataArrayB

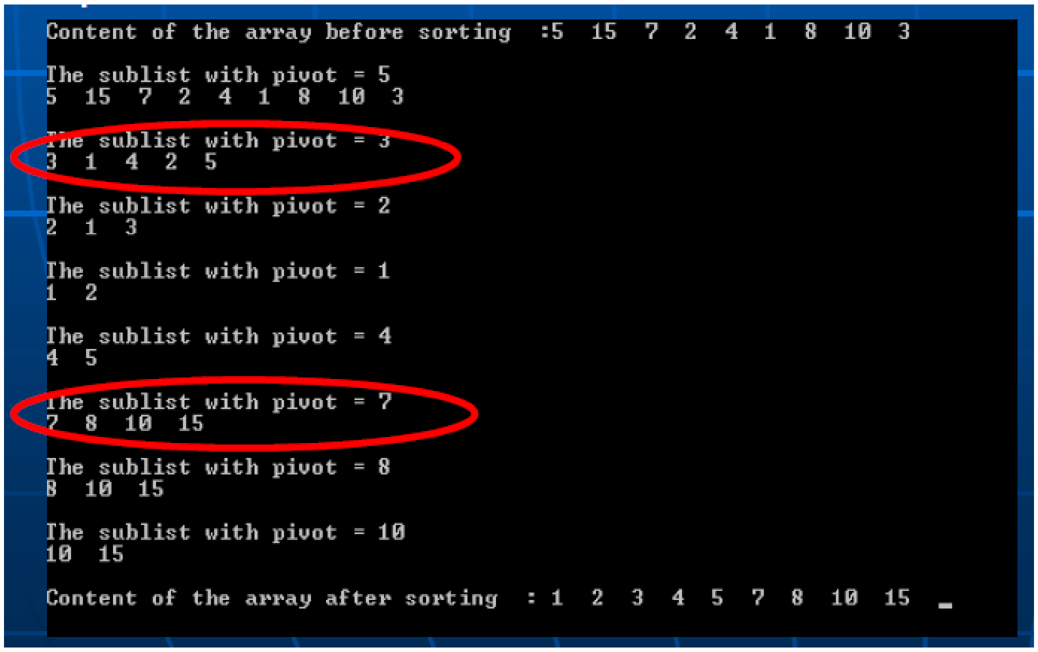
b. dataArrayA is an example of a worst case data — totally unsorted list, while dataArrayB is an example of a best case data — almost sorted list. Analyze the output displayed from each sorting technique to find the total number of passes, number of data comparisons and number of data swapping. Make conclusions on which sorting technique is the best for the worst case data and which technique is the best for best case data. Which technique has performance that does not depend on the initial arrangement of data?

c. Fill in the following table to help you discuss the performance analysis.

Technique	**Case**	**No of Comparisons**	**No of Swaps**	**No of Passes**
Conventional Bubble Sort	Worst Case			
	Best Case			
Improved Bubble Sort	Worst Case			
	Best Case			
Selection Sort	Worst Case			
	Best Case			
Insertion Sort	Worst Case			
	Best Case			

2. Write a C++ program that implements the Quick Sort algorithm to sort a list of integers [5 15 7 2 4 1 8 10 3].
   Your program must display the tracing output exactly in the format shown below, including:
   a) Unsorted data
   b) Content of partitioned sublists showing
      - pivot value
      - elements in the sublist
   c) Content of sublists processed inrecursive cals, showing
      - elements in the sublist
   d) Final sorted data

   The expected output format is illustrated as follows (your program must produce the same style of output from the slide):



```
Content of the array before sorting  :5   15   7   2   4   1   8   10   3

The sublist with pivot = 5
5   15   7   2   4   1   8   10   3

The sublist with pivot = 3
3   1   4   2   5

The sublist with pivot = 2
2   1   3

The sublist with pivot = 1
1   2

The sublist with pivot = 4
4   5

The sublist with pivot = 7
7   8   10   15

The sublist with pivot = 8
8   10   15

The sublist with pivot = 10
10   15

Content of the array after sorting  : 1   2   3   4   5   7   8   10   15
```