

Name: **Muhammad Anas Baig**Enrollment No.: **01-134152-037**Section: **BS(CS)-4A****LAB-JOURNAL-10****Exercise 1:**

Implement the following sorting algorithms using a separate function for each. Each function should accept an array of integers and the size of the array and sort the array using the respective algorithm. Also provide a function display() to print the array elements.

- Selection Sort
- Bubble Sort
- Insertion Sort

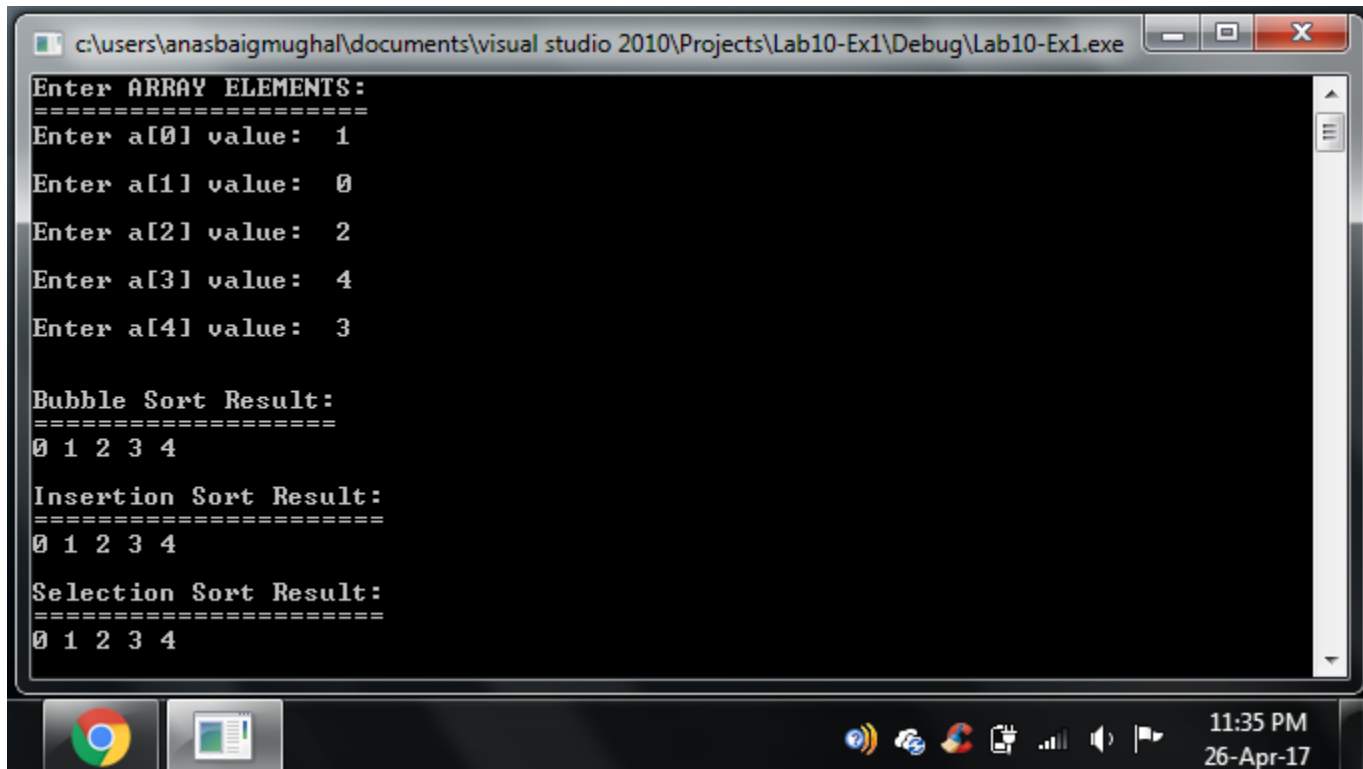
Solution:**Main.cpp File:**

```
1. #include <iostream>
2. #include "conio.h"
3. using namespace std;
4.
5. void selectionSort(int arr[], int size)
6. {
7.     for( int i=0; i<size; i++ )
8.     {
9.         int smallPos = i;
10.        int smallest = arr[i];
11.
12.        for( int j=i+1; j<size; j++ )
13.        {
14.            if( arr[j]<smallest )
15.            {
16.                smallPos = j;
17.                smallest = arr[smallPos];
18.            }
19.        }
20.        arr[smallPos] = arr[i];
21.        arr[i] = smallest;
22.    }
23. }
24.
25. void bubbleSort(int arr[], int size)
26. {
27.     for( int i=0; i<(size-1); i++ )
28.     {
29.         for( int j=0; j<(size-1); j++ )
```

```
30.     {
31.         if( arr[j] > arr[j+1] )
32.         {
33.             int temp = arr[j];
34.             arr[j] = arr[j+1];
35.             arr[j+1] = temp;
36.         }
37.     }
38. }
39. }
40.
41. void insertionSort(int arr[], int size)
42. {
43.     for( int i=1; i<(size-1); i++ )
44.     {
45.         int temp=arr[i];
46.         int j=i-1;
47.
48.         while((temp<arr[j])&&(j>=0))
49.         {
50.             arr[j+1]=arr[j];    //moves element forward
51.             j=j-1;
52.         }
53.
54.         arr[j+1]=temp;
55.     }
56. }
57.
58. int main()
59. {
60.     int a[5];
61.     cout<<"Enter ARRAY ELEMENTS:"<<endl;
62.     cout<<"===== "<<endl;
63.
64.     for(int i=0; i<5; i++)
65.     {
66.         cout<<"Enter a["<<i<<" ] value:";
67.         cin>>a[i];
68.         cout<<endl;
69.     }
70.
71.     cout<<endl;
72.     cout<<"Bubble Sort Result:"<<endl;
73.     cout<<"===== "<<endl;
74.     bubbleSort(a, 5);
75.     for( int i=0; i<5; i++ )
76.     {
77.         cout<<a[i]<<" ";
78.     }
79.
80.     cout<<endl;
81.     cout<<endl;
82.     cout<<"Insertion Sort Result:"<<endl;
83.     cout<<"===== "<<endl;
84.     insertionSort(a, 5);
85.     for( int i=0; i<5; i++ )
86.     {
87.         cout<<a[i]<<" ";
88.     }
89.
90.     cout<<endl;
91.     cout<<endl;
92.     cout<<"Selection Sort Result:"<<endl;
93.     cout<<"===== "<<endl;
94.     selectionSort(a, 5);
```

```
95.     for( int i=0; i<5; i++ )
96.     {
97.         cout<<a[i]<<" ";
98.     }
99.
100.         getch();
101.     }
```

Output:



```
c:\users\anasbaigmughal\documents\visual studio 2010\Projects\Lab10-Ex1\Debug\Lab10-Ex1.exe
Enter ARRAY ELEMENTS:
=====
Enter a[0] value: 1
Enter a[1] value: 0
Enter a[2] value: 2
Enter a[3] value: 4
Enter a[4] value: 3

Bubble Sort Result:
=====
0 1 2 3 4

Insertion Sort Result:
=====
0 1 2 3 4

Selection Sort Result:
=====
0 1 2 3 4
```

Exercise 2:

For each of the algorithms implemented in Exercise 1, generate a random array of size 'N' (for given values of 'N'), and count the number exchanges and number of comparisons in each of the three algorithms. Present your findings in the given tables.

	Number of Exchanges		
Size of List 'N'	Selection Sort	Bubble Sort	Insert Sort
10			
100			
1000			
5,000			
10,000			
25,000			

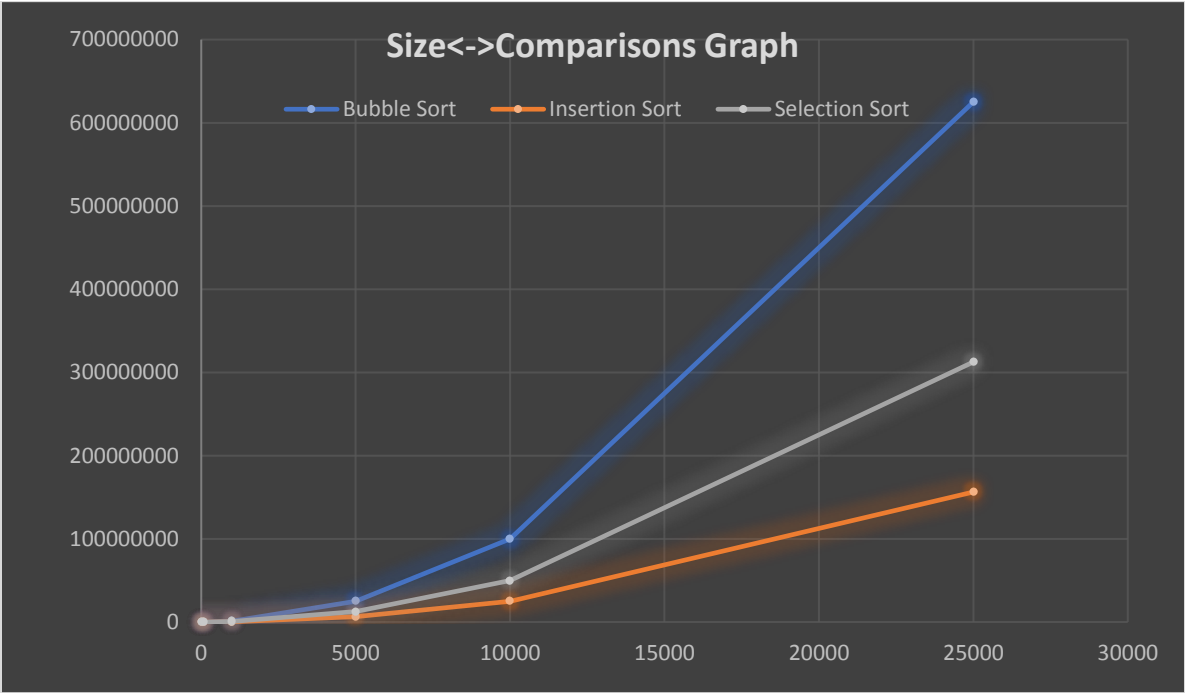
	Number of Comparison		
Size of List 'N'	Selection Sort	Bubble Sort	Insert Sort
10			
100			
1000			
5,000			
10,000			
25,000			

Also generate a line graph (in MS excel) for 'N' versus number of comparisons for each of the three algorithms. Likewise, generate a similar graph for 'N' versus number of exchanges. Find patterns in the graphs, if any.

Solution:

No. of Exchanges			
Size	Bubble Sort	Insertion Sort	Selection Sort
10	18	9	10
100	2605	99	298
1000	244828	999	5277
5000	6345554	4999	34185
10000	24898223	9999	75526
25000	156270782	24999	209777

No. of Comparisons			
Size	Bubble Sort	Insertion Sort	Selection Sort
10	81	18	45
100	9801	2605	4950
1000	998001	244828	499500
5000	24990001	6345554	12497500
10000	99980001	24898233	49995000
25000	624950001	156270782	312487500



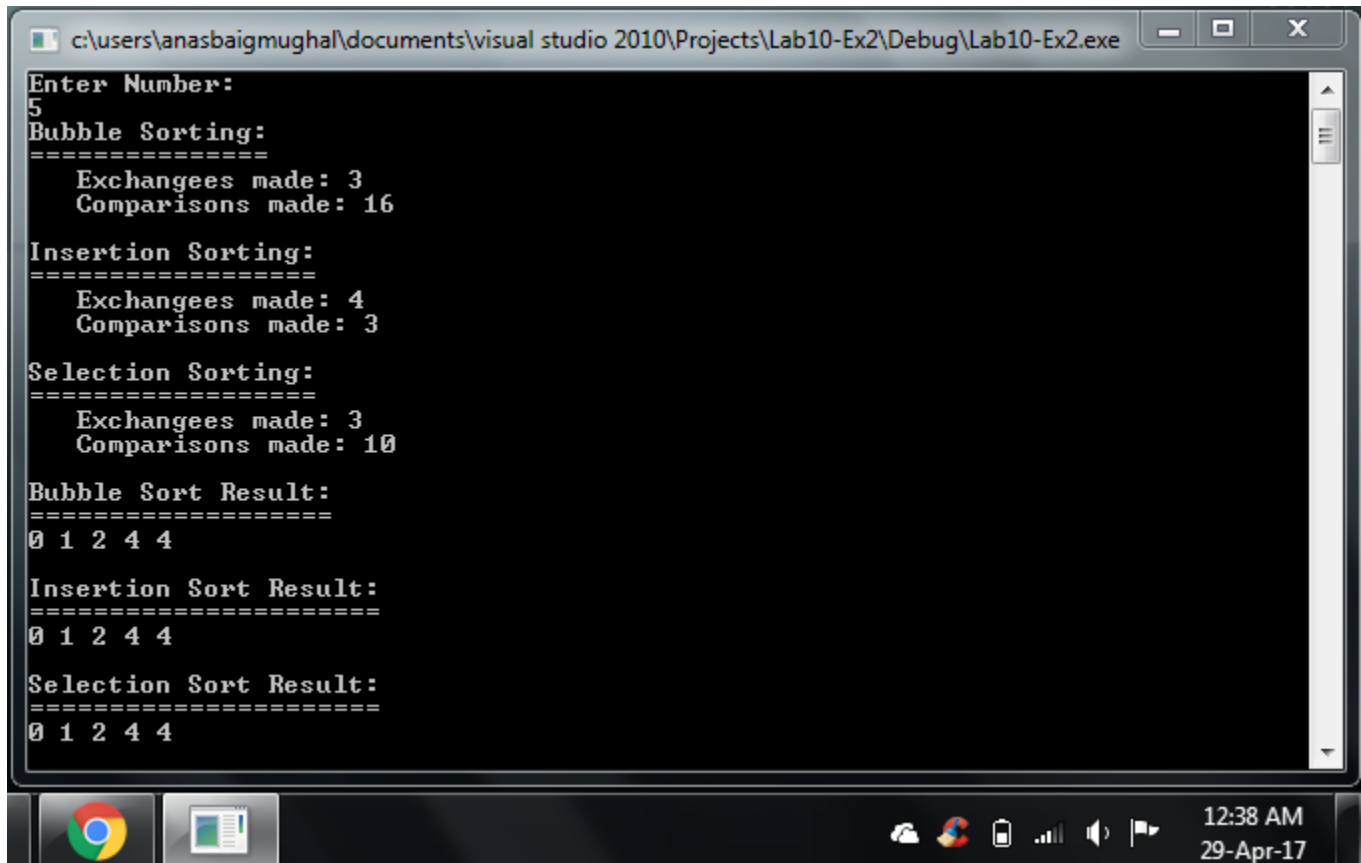
Main.cpp File:

```
1. #include <iostream>
2. #include "conio.h"
3. using namespace std;
4.
5. void selectionSort(int arr[], int size)
6. {
7.     int comparison = 0;
8.     int exchange = 0;
9.     for( int i=0; i<size; i++ )
10.    {
11.        int smallPos = i;
12.        int smallest = arr[i];
13.
14.        for( int j=i+1; j<size; j++ )
15.        {
16.            comparison++;
17.            if( arr[j]<smallest )
18.            {
19.                smallPos = j;
20.                smallest = arr[smallPos];
21.                exchange++;
22.            }
23.        }
24.        arr[smallPos] = arr[i];
25.        arr[i] = smallest;
26.    }
27.    cout<<"    Exchanges made: "<<exchange<<endl;
28.    cout<<"    Comparisons made: "<<comparison<<endl;
29. }
30.
31. void bubbleSort(int arr[], int size)
32. {
33.     int comparison = 0;
34.     int exchange = 0;
35.     for( int i=0; i<(size-1); i++ )
36.     {
37.         for( int j=0; j<(size-1); j++ )
38.         {
39.             comparison++;
40.             if( arr[j] > arr[j+1] )
41.             {
42.                 exchange++;
43.                 int temp = arr[j];
44.                 arr[j] = arr[j+1];
45.                 arr[j+1] = temp;
46.             }
47.         }
48.     }
49.     cout<<"    Exchanges made: "<<exchange<<endl;
50.     cout<<"    Comparisons made: "<<comparison<<endl;
51. }
52.
53. void insertionSort(int arr[], int size)
54. {
55.     int comparison = 0;
56.     int exchange = 0;
57.     for( int i=1; i<(size); i++ )
58.     {
59.         int temp=arr[i];
60.         int j=i-1;
61.
62.         while((temp<arr[j]) && (j>=0))
```

```
63.     {
64.         comparison++;
65.         arr[j+1]=arr[j]; //moves element forward
66.         j=j-1;
67.     }
68.
69.     exchange++;
70.     arr[j+1]=temp;
71. }
72. cout<<"   Exchanges made: "<<exchange<<endl;
73. cout<<"   Comparisons made: "<<comparison<<endl;
74. }
75.
76. int main()
77. {
78.     int n;
79.     cout<<"Enter Number:"<<endl;
80.     cin>>n;
81.     int *bubble;
82.     bubble = new int [n]; //dynamic array
83.
84.     int *insertion;
85.     insertion = new int [n]; //dynamic array
86.
87.     int *selection;
88.     selection = new int [n]; //dynamic array
89.
90.     for( int i=0; i<n; i++)
91.     {
92.         bubble[i] = insertion[i] = selection[i] = (rand() % n);
93.     }
94.
95.     cout<<"Bubble Sorting:"<<endl;
96.     cout<<"====="<<endl;
97.     bubbleSort(bubble, n);
98.     cout<<endl;
99.
100.     cout<<"Insertion Sorting:"<<endl;
101.     cout<<"====="<<endl;
102.     insertionSort(insertion, n);
103.     cout<<endl;
104.
105.     cout<<"Selection Sorting:"<<endl;
106.     cout<<"====="<<endl;
107.     selectionSort(selection, n);
108.     cout<<endl;
109.
110.     cout<<"Bubble Sort Result:"<<endl;
111.     cout<<"====="<<endl;
112.     for( int i=0; i<n; i++ )
113.     {
114.         cout<<bubble[i]<<" ";
115.     }
116.     cout<<endl<<endl;
117.
118.     cout<<"Insertion Sort Result:"<<endl;
119.     cout<<"====="<<endl;
120.     for( int i=0; i<n; i++ )
121.     {
122.         cout<<insertion[i]<<" ";
123.     }
124.     cout<<endl<<endl;
125.
126.     cout<<"Selection Sort Result:"<<endl;
127.     cout<<"====="<<endl;
```

```
128.         for( int i=0; i<n; i++ )
129.         {
130.             cout<<selection[i]<<" ";
131.         }
132.
133.         getch();
134.     }
```

Output:



```
c:\users\anasbaigmughal\documents\visual studio 2010\Projects\Lab10-Ex2\Debug\Lab10-Ex2.exe
Enter Number:
5
Bubble Sorting:
=====
    Exchanges made: 3
    Comparisons made: 16

Insertion Sorting:
=====
    Exchanges made: 4
    Comparisons made: 3

Selection Sorting:
=====
    Exchanges made: 3
    Comparisons made: 10

Bubble Sort Result:
=====
0 1 2 4 4

Insertion Sort Result:
=====
0 1 2 4 4

Selection Sort Result:
=====
0 1 2 4 4
```


Exercise 3:

Consider the following algorithm to sort a linked list using Bubble Sort.

```

BUBBLE SORT
INPUT: Pointer 'START' to head node of linked list

BOOL Swapped = FALSE
LastPTR= NULL;

DO {
    Swapped = FALSE
    PTR = START
    WHILE (PTR->next != LastPTR)
    {
        IF (PTR->data > PTR->next->data)
        {
            swap(PTR, PTR->next);
            Swapped = TRUE;
        }
        PTR = PTR->next;
    }
    LastPTR = PTR;
}
WHILE (Swapped);

```

Convert the given algorithm into a working C++ program to sort a linked list of integers.

Solution:**Node.h File:**

```

1. #pragma once
2. class node
3. {
4. public:
5.     int data;
6.     node *next;
7. public:
8.     node(void);
9. };

```

Node.cpp File:

```

1. #include "node.h"
2.
3. node::node(void)
4. {
5. }

```

List.h File:

```

1. #include "node.h"
2.

```

```
3. #pragma once
4. class list
5. {
6. public:
7.     node *head;
8. public:
9.     list(void);
10.    bool isEmpty();
11.    void addNode(int);
12.    void display();
13. };
```

List.cpp File:

```
1. #include "list.h"
2. #include <math.h>
3. #include <stdio.h>
4. #include "node.h"
5. #include <iostream>
6. using namespace std;
7.
8. list::list(void)
9. {
10.     head = NULL;
11. }
12.
13. bool list::isEmpty()
14. {
15.     if( head == NULL ) //condition for empty list
16.     {
17.         return true;
18.     }
19.     else
20.     {
21.         return false;
22.     }
23. }
24.
25. void list::addNode(int value) //logic of addNode() is based on 'insert at end'
26. {
27.     node *ptr = new node; //node creation
28.     ptr->data = 0;
29.     ptr->next = NULL;
30.
31.     ptr->data = value; //assigning values
32.
33.     if( isEmpty() ) //empty list
34.     {
35.         head = ptr;
36.     }
37.     else
38.     {
39.         ptr->next = head;
40.         head = ptr;
41.     }
42. }
43.
44. void list::display() //function to display list
45. {
46.     node *temp = head;
47.
48.     while ( temp != NULL ) //traverse until reached at end
49.     {
50.         cout<<temp->data<<" ";
```

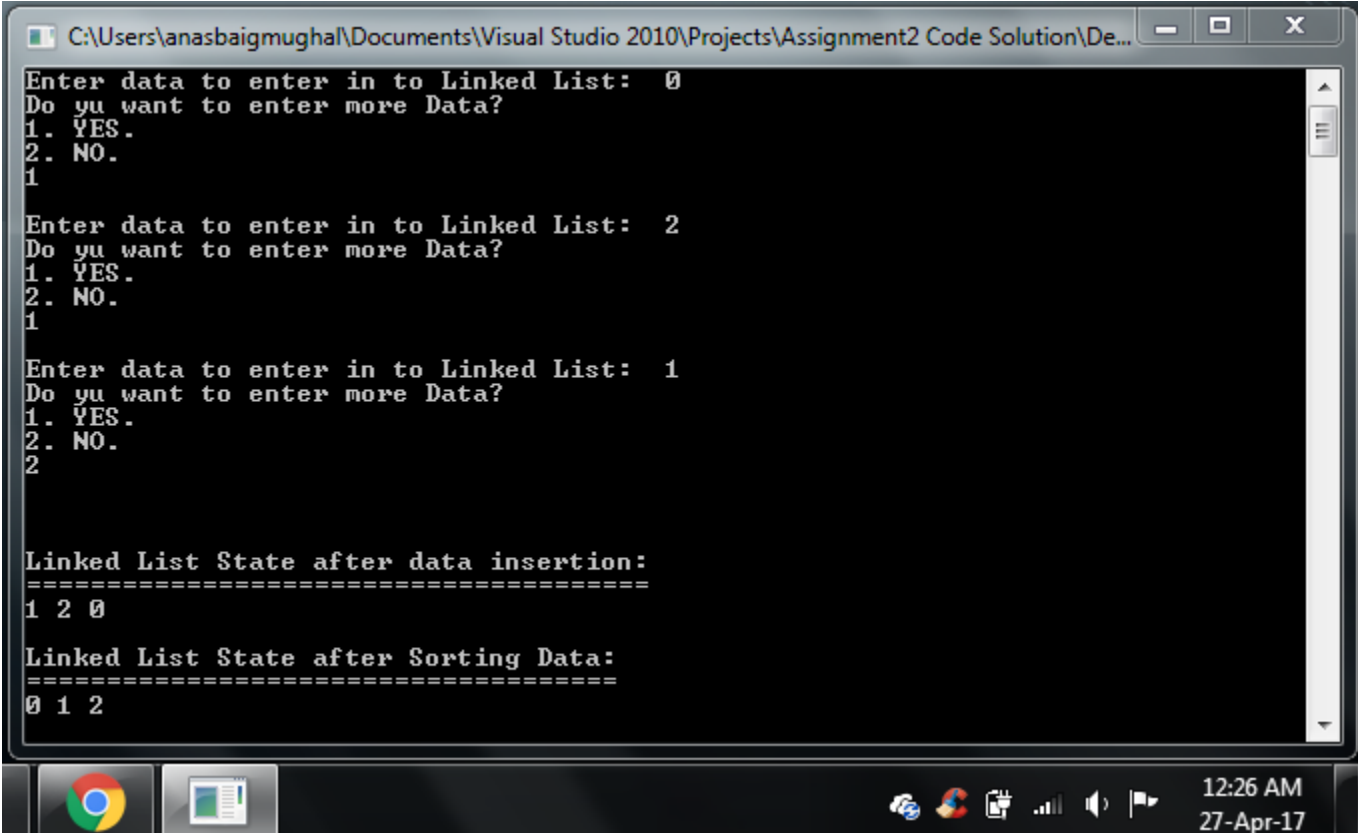
```
51.     temp = temp->next;
52. }
53. }
```

Main.cpp File:

```
1. #include "list.h"
2. #include "node.h"
3. #include "conio.h"
4. #include <iostream>
5. using namespace std;
6.
7. void _swap(node * n1, node * n2)
8. {
9.     int temp;
10.
11.     temp = n1->data;
12.     n1->data = n2->data;
13.     n2->data = temp;
14. }
15.
16. void bubbleSort( node *head )
17. {
18.     bool swapped = false;
19.     node *lastPtr = NULL;
20.     node *temp;
21.     do
22.     {
23.         swapped = false;
24.         temp = head;
25.         while( temp->next != lastPtr )
26.         {
27.             if( temp->data > temp->next->data )
28.             {
29.                 _swap( temp, temp->next);
30.                 swapped = true;
31.             }
32.             temp = temp->next;
33.         }
34.         lastPtr = temp;
35.     }
36.     while( swapped );
37. }
38.
39. void main()
40. {
41.     list l;
42.     int data;
43.     int choice;
44.
45.     do
46.     {
47.         cout<<"Enter data to enter in to Linked List: ";
48.         cin>>data;
49.         l.addNode(data);
50.
51.         cout<<"Do you want to enter more Data?"<<endl;
52.         cout<<"1. YES."<<endl;
53.         cout<<"2. NO."<<endl;
54.         cin>>choice;
55.         cout<<endl;
56.     }
57.     while( choice == 1 );
58. }
```

```
59.     cout<<endl;
60.     cout<<endl;
61.     cout<<"Linked List State after data insertion:"<<endl;
62.     cout<<"===== "<<endl;
63.     l.display();
64.
65.     cout<<endl;
66.     cout<<endl;
67.     cout<<"Linked List State after Sorting Data:"<<endl;
68.     cout<<"===== "<<endl;
69.     bubbleSort( l.head );
70.     l.display();
71.
72.     getch();
73. }
```

Output:



```
C:\Users\anasbaigmughal\Documents\Visual Studio 2010\Projects\Assignment2 Code Solution\De...
Enter data to enter in to Linked List: 0
Do yu want to enter more Data?
1. YES.
2. NO.
1

Enter data to enter in to Linked List: 2
Do yu want to enter more Data?
1. YES.
2. NO.
1

Enter data to enter in to Linked List: 1
Do yu want to enter more Data?
1. YES.
2. NO.
2

Linked List State after data insertion:
=====
1 2 0

Linked List State after Sorting Data:
=====
0 1 2
```