

Name: Muhammad Anas BaigEnrollment No.: 01-134152-037Section: **BS(CS)-4A****LAB-JOURNAL-7****Exercise 1:**

Implement the class Doubly Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following.

```
class List
{
private:
    Node * head;
public:
    List();
    ~List();
    // Checks if the list is empty or not
    bool emptyList();

    // Inserts a new node with value 'newV' after the node
    // containing value 'oldV'. If a node with value 'oldV' does
    // not exist, inserts the new node at the end.
    void insertafter(int oldV, int newV);

    // Deletes the node containing the specified value
    void deleteNode(int value);

    // Inserts a new node at the start of the list
    void insert_begin(int value);

    // Inserts a new node at the end of the list
    void insert_end(int value);

    // Displays the values stored in the list
    void traverse();
};
```

Solution:**Node.h File:**

```
1. #pragma once
2. class Node
3. {
4. public:
5.     Node *prev;
6.     int data;
7.     Node *next;
8. public:
9.     Node(void);
10. };
```

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 insertAfter(int, int);
12.    void deleteNode(int);
13.    void insertBegin(int);
14.    void insertEnd(int);
15.    void display();
16. };
```

List.cpp File:

```
1. #include "List.h"
2. #include "Node.h"
3. #include <iostream>
4. using namespace std;
5.
6. List::List(void)
7. {
8.     head = '\0';
9. }
10.
11. bool List::isEmpty()
12. {
13.     if( head == '\0' )
14.     {
15.         return true;
16.     }
17.     else
18.     {
19.         return false;
20.     }
```

```
21. }
22.
23. void List::insertAfter(int oldVal, int newVal)
24. {
25.     if( !isEmpty() )
26.     {
27.         Node *ptr = new Node;
28.         ptr->prev = '\0';
29.         ptr->data = 0;
30.         ptr->next = '\0';
31.
32.         Node *temp = head;
33.         while( temp->data != oldVal && temp != '\0' )
34.         {
35.             temp = temp->next;
36.         }
37.
38.         ptr->data = newVal;
39.         ptr->next = temp->next;
40.         ptr->prev = temp;
41.         temp->next->prev = ptr;
42.         temp->next = ptr;
43.     }
44.     else
45.     {
46.         cout<<"SORRY!!! List is Empty."<<endl;
47.     }
48. }
49.
50. void List::deleteNode(int value)
51. {
52.     if( !isEmpty() )
53.     {
54.         Node *temp = head;
55.
56.         while( temp->data != value && temp != '\0' )
57.         {
58.             temp = temp->next;
59.         }
60.
61.         if( temp == '\0' ) //value not found
62.         {
63.             cout<<"SORRY!!! Value not found in list."<<endl;
64.         }
65.         else //value is found
66.         {
67.             if( temp->prev == '\0' && temp->next == '\0' ) //list has one node only and that first node contains data
68.             {
69.                 head = '\0';
70.             }
71.             else if( temp->prev == '\0' && temp->next != '\0' ) //first node contains data and it is not the only node
72.             {
73.                 head = head->next;
74.                 head->prev = '\0';
75.             }
76.             else if( temp->prev != '\0' && temp->next == '\0' ) //last node contains data and it is not the only node
77.             {
78.                 temp->prev->next = '\0';
79.             }
80.             else
81.             {
82.                 temp->prev->next = temp->next;
83.                 temp->next->prev = temp->prev;
84.             }
85.             delete temp;
86.         }
87.     }
```

```
88.     else
89.     {
90.         cout<<"SORRY!!! List is Empty."<<endl;
91.     }
92. }
93.
94. void List::insertBegin(int newVal)
95. {
96.     Node *ptr = new Node;
97.     ptr->prev = '\0';
98.     ptr->data = 0;
99.     ptr->next = '\0';
100.
101.     if( !isEmpty() )
102.     {
103.         ptr->data = newVal;
104.         ptr->next = head;
105.         head->prev = ptr;
106.         head = ptr;
107.     }
108.     else
109.     {
110.         ptr->data = newVal;
111.         head = ptr;
112.     }
113. }
114.
115. void List::insertEnd(int newVal)
116. {
117.     Node *ptr = new Node;
118.     ptr->prev = '\0';
119.     ptr->data = 0;
120.     ptr->next = '\0';
121.
122.     if( !isEmpty() )
123.     {
124.
125.         Node *temp = head;
126.
127.         while( temp->next != '\0' )
128.         {
129.             temp = temp->next;
130.         }
131.
132.         ptr->data = newVal;
133.         ptr->prev = temp;
134.         temp->next = ptr;
135.     }
136.     else
137.     {
138.         ptr->data = newVal;
139.         head = ptr;
140.     }
141. }
142.
143. void List::display()
144. {
145.     if( !isEmpty() )
146.     {
147.         Node *temp = head;
148.
149.         while( temp != '\0' )
150.         {
151.             cout<<temp->data<<" ";
152.             temp = temp->next;
153.         }
154.         cout<<endl;
155.     }
156.     else
157.     {
```

```
158.         cout<<"SORRY!!! List is Empty."<<endl;
159.     }
160. }
```

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 main()
8. {
9.     List l;
10.
11.     cout<<"Doubly Linked List Empty Check:"<<endl;
12.     cout<<"====="<<endl;
13.     l.display();
14.     cout<<endl;
15.
16.     l.insertBegin(1);
17.     l.insertBegin(2);
18.     l.insertBegin(3);
19.     l.insertBegin(4);
20.     l.insertBegin(5);
21.     cout<<"Doubly Linked List state after Beginning Insertion:"<<endl;
22.     cout<<"====="<<endl;
23.     l.display();
24.     cout<<endl;
25.
26.     l.insertAfter(3,6);
27.     cout<<"Doubly Linked List state after Mid Insertion:"<<endl;
28.     cout<<"====="<<endl;
29.     l.display();
30.     cout<<endl;
31.
32.     l.insertEnd(100);
33.     cout<<"Doubly Linked List state after End Insertion:"<<endl;
34.     cout<<"====="<<endl;
35.     l.display();
36.     cout<<endl;
37.
38.     l.deleteNode(3);
39.     cout<<"Doubly Linked List state after Node Deletion:"<<endl;
40.     cout<<"====="<<endl;
41.     l.display();
42.     cout<<endl;
43.
44.     getch();
45. }
```

Output:

```

c:\users\muhammad anas baig\documents\visual studio 2010\Projects\Lab7-Ex1\Debug\Lab7-Ex1...
Doubly Linked List Empty Check:
=====
SORRY!!! List is Empty.

Doubly Linked List state after Beginning Insertion:
=====
5 4 3 2 1

Doubly Linked List state after Mid Insertion:
=====
5 4 3 6 2 1

Doubly Linked List state after End Insertion:
=====
5 4 3 6 2 1 100

Doubly Linked List state after Node Deletion:
=====
5 4 6 2 1 100

```

Exercise 2:

Implement the Stack using a doubly linked list. Provide the standard push, pop and top operations in the class.

Solution:**Node.h File:**

```

1. #pragma once
2. class Node
3. {
4. public:
5.     Node *prev;
6.     int data;
7.     Node *next;
8. public:
9.     Node(void);
10. };

```

Node.cpp File:

```

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

```

doublyListStack.h File:

```

1. #include "Node.h"
2.
3. #pragma once

```

```
4. class doublyListStack
5. {
6. public:
7.     Node *top;
8. public:
9.     doublyListStack(void);
10.    bool isEmpty();
11.    void push(int);
12.    int pop();
13.    int getFront();
14.    void display();
15. };
```

doublyListStack.cpp File:

```
1. #include "doublyListStack.h"
2. #include "Node.h"
3. #include <iostream>
4. using namespace std;
5.
6. doublyListStack::doublyListStack(void)
7. {
8.     top = '\0';
9. }
10.
11. bool doublyListStack::isEmpty()
12. {
13.     if( top == '\0' )
14.     {
15.         return true;
16.     }
17.     else
18.     {
19.         return false;
20.     }
21. }
22.
23. void doublyListStack::push(int newVal)
24. {
25.     Node *ptr = new Node;
26.     ptr->prev = '\0';
27.     ptr->data = 0;
28.     ptr->next = '\0';
29.
30.     if( !isEmpty() )
31.     {
32.         ptr->data = newVal;
33.         ptr->next = top;
34.         top->prev = ptr;
35.         top = ptr;
36.     }
37.     else
38.     {
39.         ptr->data = newVal;
40.         top = ptr;
41.     }
42. }
43.
44. int doublyListStack::pop()
45. {
46.     if( !isEmpty() )
47.     {
48.         Node *temp = top;
49.         int tempData = top->data;
50.
51.         if( top->next == '\0' )
52.         {
53.             top = '\0';
54.         }
```

```

55.         else
56.         {
57.             top = top->next;
58.             top->prev = '\0';
59.         }
60.         return (tempData);
61.     }
62.     else
63.     {
64.         cout<<"SORRY!!! List is Empty."<<endl;
65.         return (-1);
66.     }
67. }
68.
69. int doublyListStack::getFront()
70. {
71.     if( !isEmpty() )
72.     {
73.         return (top->data);
74.     }
75.     else
76.     {
77.         cout<<"SORRY!!! List is Empty."<<endl;
78.     }
79. }
80.
81. void doublyListStack::display()
82. {
83.     if( !isEmpty() )
84.     {
85.         Node *temp = top;
86.
87.         while( temp != '\0' )
88.         {
89.             cout<<temp->data<<" ";
90.             temp = temp->next;
91.         }
92.         cout<<endl;
93.     }
94.     else
95.     {
96.         cout<<"SORRY!!! List is Empty."<<endl;
97.     }
98. }

```

Main.cpp File:

```

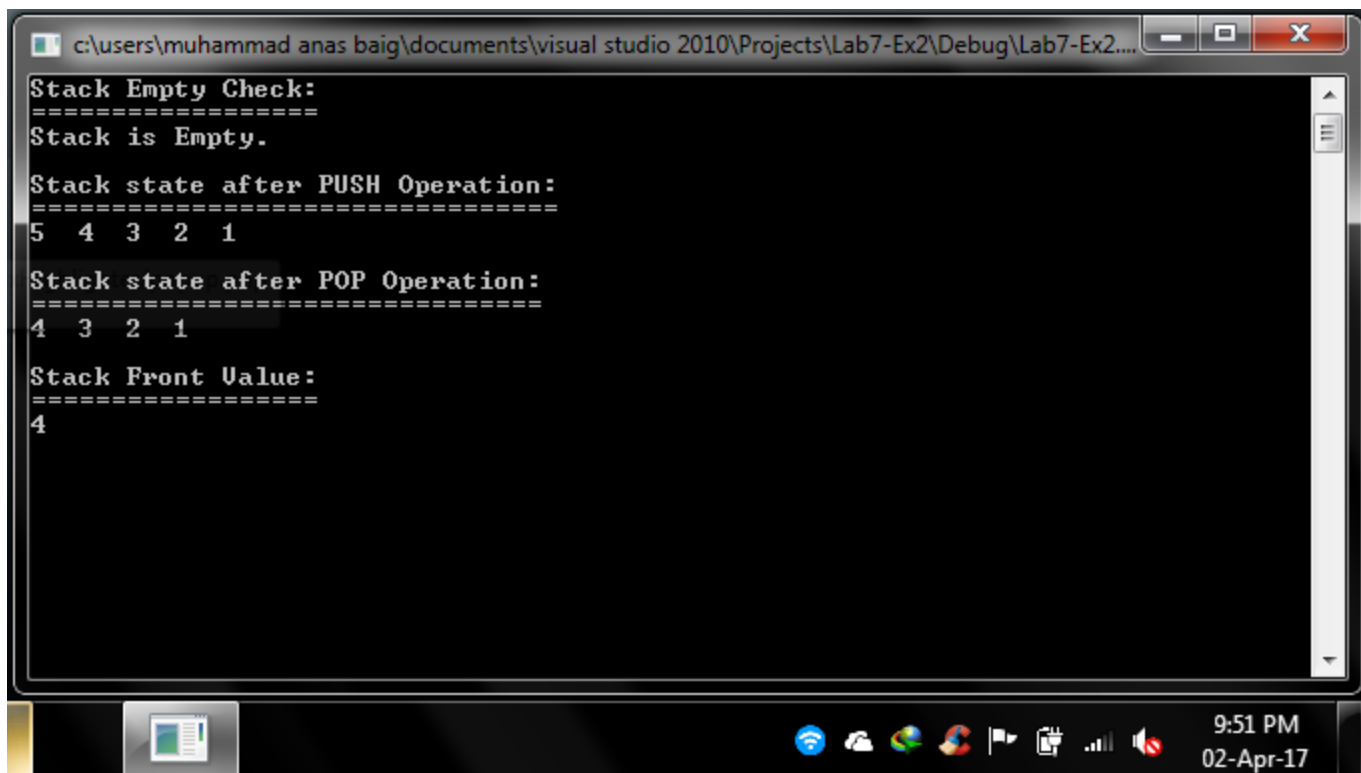
1. #include "doublyListStack.h"
2. #include "Node.h"
3. #include "conio.h"
4. #include <iostream>
5. using namespace std;
6.
7. void main()
8. {
9.     doublyListStack l;
10.
11.     cout<<"Stack Empty Check:"<<endl;
12.     cout<<"======"<<endl;
13.     if( l.isEmpty() )
14.     {
15.         cout<<"Stack is Empty."<<endl;
16.     }
17.     else
18.     {
19.         cout<<"Stack is not Empty."<<endl;
20.     }
21.     cout<<endl;
22.

```



```
23.     l.push(1);
24.     l.push(2);
25.     l.push(3);
26.     l.push(4);
27.     l.push(5);
28.     cout<<"Stack state after PUSH Operation:"<<endl;
29.     cout<<"===== "<<endl;
30.     l.display();
31.     cout<<endl;
32.
33.     l.pop();
34.     cout<<"Stack state after POP Operation:"<<endl;
35.     cout<<"===== "<<endl;
36.     l.display();
37.     cout<<endl;
38.
39.     cout<<"Stack Front Value:"<<endl;
40.     cout<<"===== "<<endl;
41.     cout<<l.getFront();
42.     cout<<endl;
43.
44.     getch();
45. }
```

Output:



```
c:\users\muhammad anas baig\documents\visual studio 2010\Projects\Lab7-Ex2\Debug\Lab7-Ex2...
Stack Empty Check:
=====
Stack is Empty.

Stack state after PUSH Operation:
=====
5 4 3 2 1

Stack state after POP Operation:
=====
4 3 2 1

Stack Front Value:
=====
4
```

Exercise 3:

Write a program that stores student ID, name and age in a doubly linked list. Also, write a function to search and display record(s) based on student name.

Solution:

Student.h File:

```
1. #include <string>
2. using namespace std;
3.
4. #pragma once
5. class student
6. {
7. public:
8.     int id;
9.     string name;
10.    int age;
11.    student *prev;
12.    student *next;
13. public:
14.    student(void);
15. };
```

Student.cpp File:

```
1. #include "student.h"
2. #include <string>
3. #include <iostream>
4. using namespace std;
5.
6. student::student(void)
7. {
8. }
```

studentList.h File:

```
1. #include "student.h"
2. #include <string>
3. using namespace std;
4.
5. #pragma once
6. class studentList
7. {
8. public:
9.     student *head;
10. public:
11.     studentList(void);
12.     bool isEmpty();
13.     void addStudent(int, string, int);
14.     void findStudent(string);
15.     void display();
16. };
```

studentList.cpp File:

```
1. #include "studentList.h"
2. #include "student.h"
3. #include <string>
4. #include <iostream>
5. using namespace std;
6.
```

```
7. studentList::studentList(void)
8. {
9.     head = '\0';
10. }
11.
12. bool studentList::isEmpty()
13. {
14.     if(head == '\0')
15.     {
16.         return true;
17.     }
18.     else
19.     {
20.         return false;
21.     }
22. }
23.
24. void studentList::addStudent(int id, string name, int age)
25. {
26.     student *ptr = new student;
27.     ptr->id = 0;
28.     ptr->name = '\0';
29.     ptr->age = 0;
30.     ptr->prev = '\0';
31.     ptr->next = '\0';
32.
33.     ptr->id = id;
34.     ptr->name = name;
35.     ptr->age = age;
36.
37.     if(!isEmpty())
38.     {
39.         ptr->next = head;
40.         head->prev = ptr;
41.         head = ptr;
42.     }
43.     else
44.     {
45.         head=ptr;
46.     }
47. }
48.
49. void studentList::findStudent(string name)
50. {
51.     if(!isEmpty())
52.     {
53.         student *temp;
54.         temp = head;
55.
56.         while(temp->name != name && temp!= '\0')
57.         {
58.             temp = temp->next;
59.         }
60.
61.         cout<<"Student Record:"<<endl;
62.         cout<<"======"<<endl;
63.         cout<<"Student ID: "<<temp->id<<endl;
64.         cout<<"Student Name: "<<temp->name<<endl;
65.         cout<<"Student Age: "<<temp->age<<endl;
66.     }
67.     else
68.     {
69.         cout<<"SORRY!!! Student List is Empty"<<endl;
70.     }
71. }
72.
73. void studentList::display()
74. {
75.     if(!isEmpty())
76.     {
```

```

77.     student *temp;
78.     temp = head;
79.
80.     cout<<"Student Record List:"<<endl;
81.     cout<<"======"<<endl;
82.     while(temp != '\0')
83.     {
84.         cout<<"Student ID: "<<temp->id<<endl;
85.         cout<<"Student Name: "<<temp->name<<endl;
86.         cout<<"Student Age: "<<temp->age<<endl;
87.         cout<<endl;
88.         temp = temp->next;
89.     }
90. }
91. else
92. {
93.     cout<<"SORRY!!! Student List is Empty"<<endl;
94. }
95. }

```

Main.cpp File:

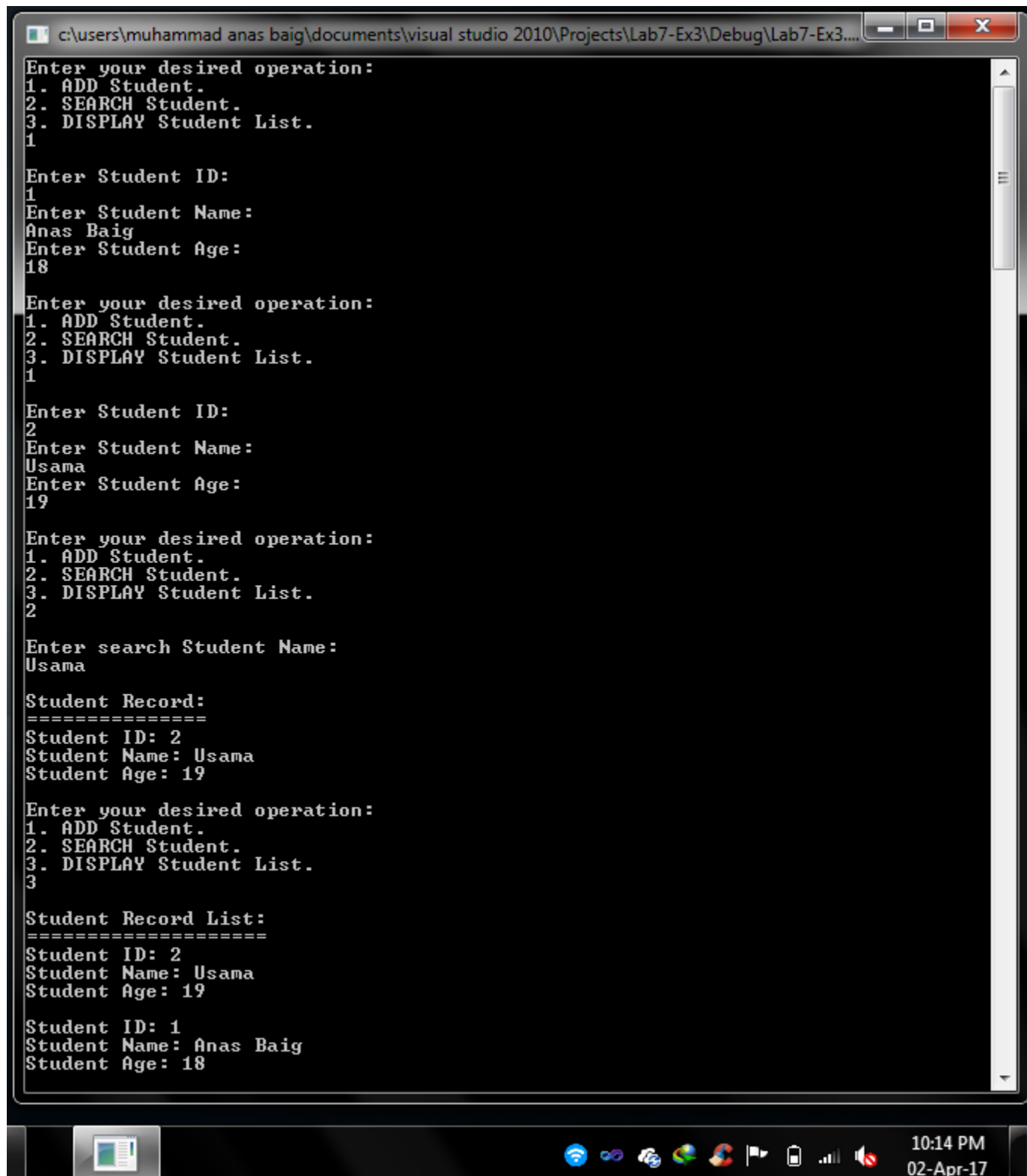
```

1. #include "studentList.h"
2. #include <string>
3. #include "student.h"
4. #include "conio.h"
5. #include <iostream>
6. using namespace std;
7.
8. void main()
9. {
10.     studentList l;
11.     int choice;
12.     int id;
13.     string searchName;
14.     string name;
15.     int age;
16.
17.     do
18.     {
19.         cout<<"Enter your desired operation:"<<endl;
20.         cout<<"1. ADD Student."<<endl;
21.         cout<<"2. SEARCH Student."<<endl;
22.         cout<<"3. DISPLAY Student List."<<endl;
23.         cin>>choice;
24.         cout<<endl;
25.         if(choice == 1)
26.         {
27.             cout<<"Enter Student ID:"<<endl;
28.             cin>>id;
29.             cout<<"Enter Student Name:"<<endl;
30.             cin.ignore(); //getline is having issue in while loop so that this statement is used
31.             getline(cin, name);
32.             cout<<"Enter Student Age:"<<endl;
33.             cin>>age;
34.             l.addStudent(id, name, age);
35.             cout<<endl;
36.         }
37.         else if(choice == 2)
38.         {
39.             cout<<"Enter search Student Name:"<<endl;
40.             cin>>searchName;
41.             cout<<endl;
42.             l.findStudent(searchName);
43.             cout<<endl;
44.         }
45.         else
46.         {
47.             l.display();

```

```
48.         cout<<endl;
49.     }
50. }
51. while(choice == 1 || choice == 2 || choice == 3); //1 for ADD STUDENT, 2 for SEARCH STUDENT, 3 for DISPLAY ST
UDENT LIST
52.     getch();
53. }
```

Output:



```
c:\users\muhammad anas baig\documents\visual studio 2010\Projects\Lab7-Ex3\Debug\Lab7-Ex3....
Enter your desired operation:
1. ADD Student.
2. SEARCH Student.
3. DISPLAY Student List.
1

Enter Student ID:
1
Enter Student Name:
Anas Baig
Enter Student Age:
18

Enter your desired operation:
1. ADD Student.
2. SEARCH Student.
3. DISPLAY Student List.
2

Enter Student ID:
2
Enter Student Name:
Usama
Enter Student Age:
19

Enter your desired operation:
1. ADD Student.
2. SEARCH Student.
3. DISPLAY Student List.
2

Enter search Student Name:
Usama

Student Record:
=====
Student ID: 2
Student Name: Usama
Student Age: 19

Enter your desired operation:
1. ADD Student.
2. SEARCH Student.
3. DISPLAY Student List.
3

Student Record List:
=====
Student ID: 2
Student Name: Usama
Student Age: 19

Student ID: 1
Student Name: Anas Baig
Student Age: 18
```