Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **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:

```
#pragma once
class Node
{
public:
    Node *prev;
    int data;
    Node *next;
public:
    Node(void);
};
```

Node.cpp File:

```
#include "Node.h"

Node::Node(void)
{
}
```

List.h File:

```
#include "Node.h"

#pragma once
class List
{
public:
    Node *head;
public:
    List(void);
    bool isEmpty();
    void insertAfter(int, int);
    void deleteNode(int);
    void insertBegin(int);
    void insertEnd(int);
    void display();
};
```

List.cpp File:

```
#include "List.h"
#include "Node.h"
#include <iostream>
using namespace std;

List::List(void)
{
    head = '\0';
}

bool List::isEmpty()
{
    if( head == '\0' )
    {
        return true;
    }
    else
    {
        return false;
    }
```

```
}

void List::insertAfter(int oldVal, int newVal)
{
    if( !isEmpty() )
    {
        Node *ptr = new Node;
        ptr->prev = '\0';
        ptr->data = 0;
        ptr->next = '\0';

        Node *temp = head;
        while( temp->data != oldVal && temp != '\0' )
        {
            temp = temp->next;
        }

        ptr->data = newVal;
        ptr->next = temp->next;
        ptr->prev = temp;
        temp->next->prev = ptr;
        temp->next = ptr;
    }
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
    }
}

void List::deleteNode(int value)
{
    if( !isEmpty() )
    {
        Node *temp = head;

        while( temp->data != value && temp != '\0' )
        {
            temp = temp->next;
        }

        if( temp == '\0' ) //value not found
        {
            cout<<"SORRY!!! Value not found in list."<<endl;
        }
        else //value is found
        {
            if( temp->prev == '\0' && temp->next == '\0' ) //list has one node only and that first node contains data
            {
                head = '\0';
            }
            else if( temp->prev == '\0' && temp->next != '\0' ) //first node contains data and it is not the only node
            {
                head = head->next;
                head->prev = '\0';
            }
            else if( temp->prev != '\0' && temp->next == '\0' ) //last node contains data and it is not the only node
            {
                temp->prev->next = '\0';
            }
            else
            {
                temp->prev->next = temp->next;
                temp->next->prev = temp->prev;
            }
        delete temp;
        }
    }
```

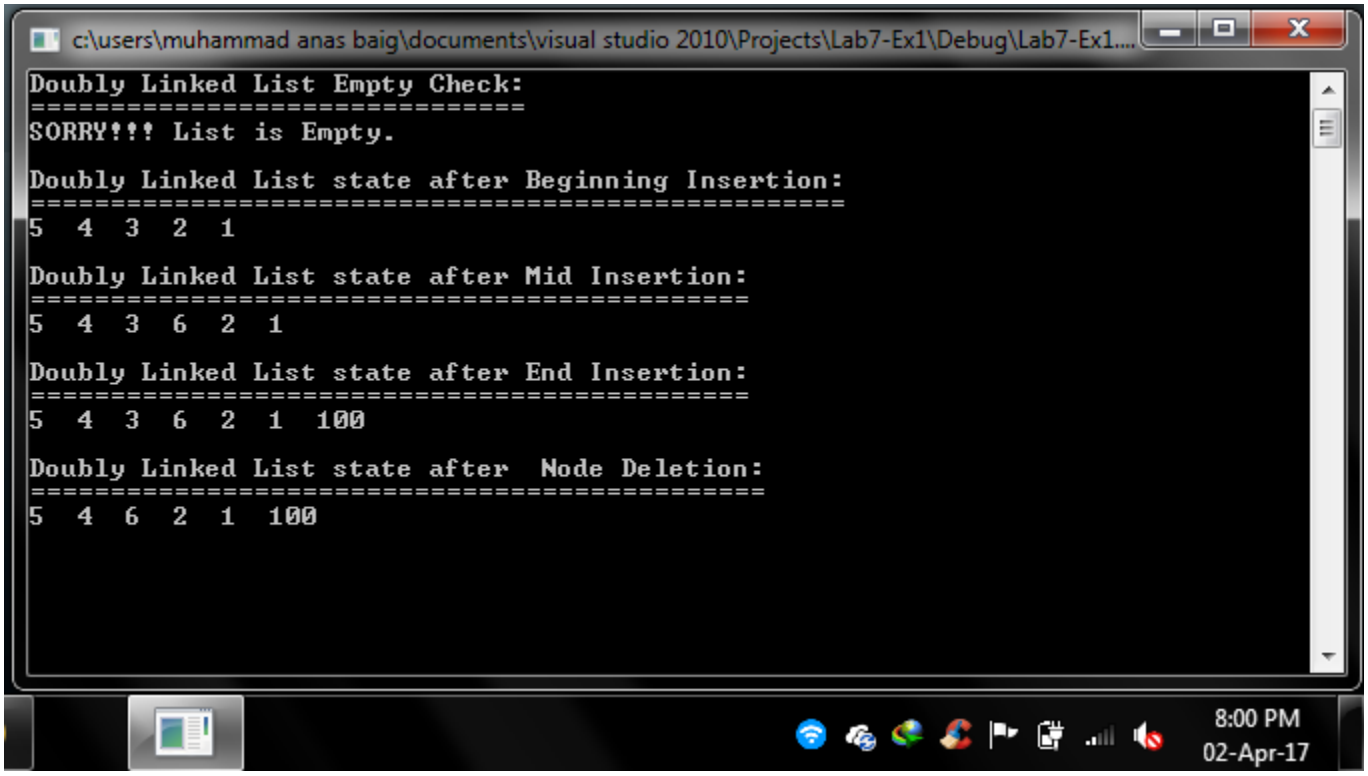
```
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
    }
}

void List::insertBegin(int newVal)
{
    Node *ptr = new Node;
    ptr->prev = '\0';
    ptr->data = 0;
    ptr->next = '\0';

        if( !isEmpty() )
        {
            ptr->data = newVal;
            ptr->next = head;
            head->prev = ptr;
            head = ptr;
        }
        else
        {
            ptr->data = newVal;
            head = ptr;
        }
    }

    void List::insertEnd(int newVal)
    {
            Node *ptr = new Node;
            ptr->prev = '\0';
            ptr->data = 0;
            ptr->next = '\0';

            if( !isEmpty() )
            {

                Node *temp = head;

                while( temp->next != '\0' )
                {
                    temp = temp->next;
                }

                ptr->data = newVal;
                ptr->prev = temp;
                temp->next = ptr;
            }
            else
            {
                ptr->data = newVal;
                head = ptr;
            }
    }

    void List::display()
    {
        if( !isEmpty() )
        {
            Node *temp = head;

            while( temp != '\0' )
            {
                cout<<temp->data<<"  ";
                temp = temp->next;
            }
            cout<<endl;
        }
        else
        {
```

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

Main.cpp File:

```
#include "List.h"
#include "Node.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    List l;

    cout<<"Doubly Linked List Empty Check:"<<endl;
    cout<<"==============================="<<endl;
    l.display();
    cout<<endl;

    l.insertBegin(1);
    l.insertBegin(2);
    l.insertBegin(3);
    l.insertBegin(4);
    l.insertBegin(5);
    cout<<"Doubly Linked List state after Beginning Insertion:"<<endl;
    cout<<"==================================================="<<endl;
    l.display();
    cout<<endl;

    l.insertAfter(3,6);
    cout<<"Doubly Linked List state after Mid Insertion:"<<endl;
    cout<<"============================================="<<endl;
    l.display();
    cout<<endl;

    l.insertEnd(100);
    cout<<"Doubly Linked List state after End Insertion:"<<endl;
    cout<<"============================================="<<endl;
    l.display();
    cout<<endl;

    l.deleteNode(3);
    cout<<"Doubly Linked List state after  Node Deletion:"<<endl;
    cout<<"============================================="<<endl;
    l.display();
    cout<<endl;

    getch();
}
```

Output:



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:

```
#pragma once
class Node
{
public:
    Node *prev;
    int data;
    Node *next;
public:
    Node(void);
};
```

Node.cpp File:

```
#include "Node.h"

Node::Node(void)
{
}
```

doublyListStack.h File:

```
#include "Node.h"

#pragma once
```

```
class doublyListStack
{
public:
    Node *top;
public:
    doublyListStack(void);
    bool isEmpty();
    void push(int);
    int pop();
    int getFront();
    void display();
};
```

doublyListStack.cpp File:

```
#include "doublyListStack.h"
#include "Node.h"
#include <iostream>
using namespace std;

doublyListStack::doublyListStack(void)
{
    top = '\0';
}

bool doublyListStack::isEmpty()
{
    if( top == '\0' )
    {
        return true;
    }
    else
    {
        return false;
    }
}

void doublyListStack::push(int newVal)
{
    Node *ptr = new Node;
    ptr->prev = '\0';
    ptr->data = 0;
    ptr->next = '\0';

    if( !isEmpty() )
    {
        ptr->data = newVal;
        ptr->next = top;
        top->prev = ptr;
        top = ptr;
    }
    else
    {
        ptr->data = newVal;
        top = ptr;
    }
}

int doublyListStack::pop()
{
    if( !isEmpty() )
    {
        Node *temp = top;
        int tempData = top->data;

        if( top->next == '\0' )
        {
            top = '\0';
        }
```

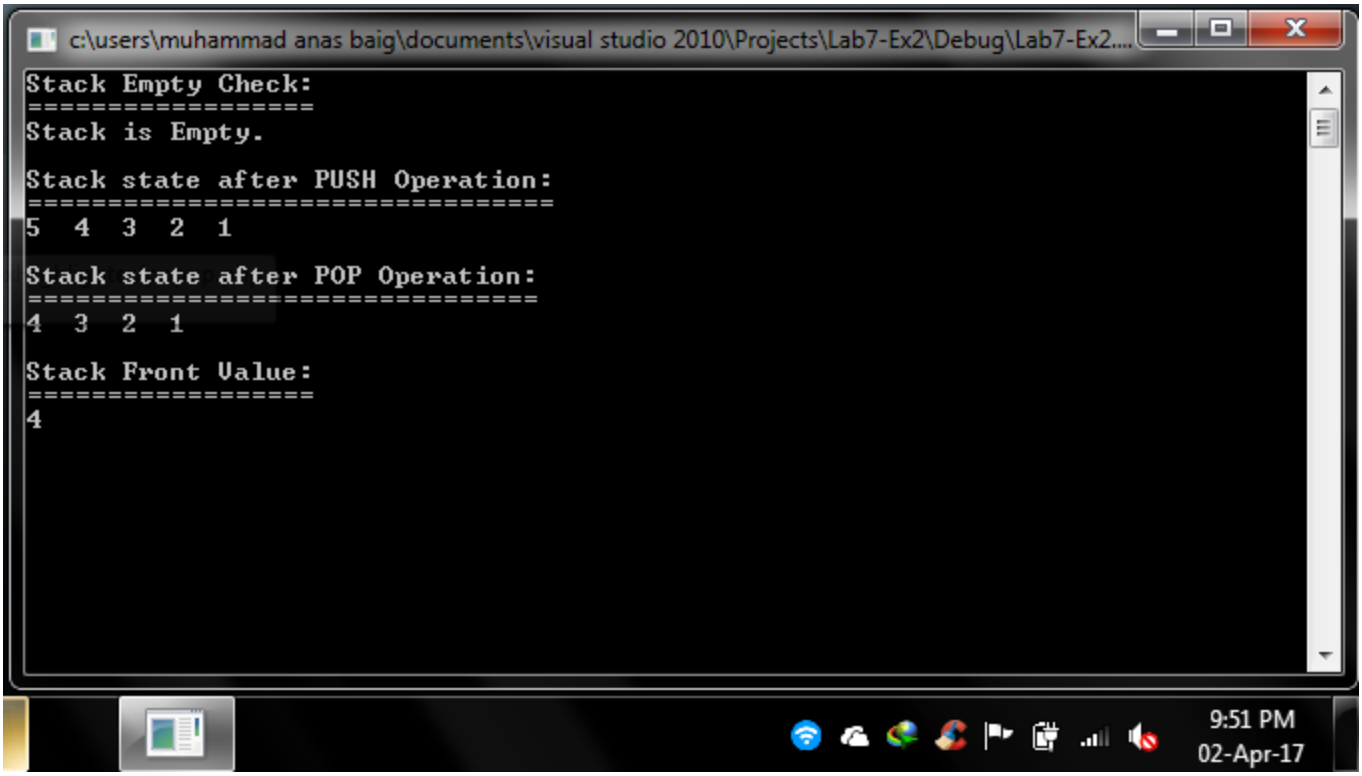
```
        else
        {
            top = top->next;
            top->prev = '\0';
        }
        return (tempData);
    }
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
        return (-1);
    }
}

int doublyListStack::getFront()
{
    if( !isEmpty() )
    {
        return (top->data);
    }
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
    }
}

void doublyListStack::display()
{
    if( !isEmpty() )
    {
        Node *temp = top;

        while( temp != '\0' )
        {
            cout<<temp->data<<"  ";
            temp = temp->next;
        }
        cout<<endl;
    }
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
    }
}
```

Main.cpp File:

```
#include "doublyListStack.h"
#include "Node.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    doublyListStack l;

    cout<<"Stack Empty Check:"<<endl;
    cout<<"==================="<<endl;
    if( l.isEmpty() )
    {
        cout<<"Stack is Empty."<<endl;
    }
    else
    {
        cout<<"Stack is not Empty."<<endl;
    }
    cout<<endl;

```

```
    l.push(1);
    l.push(2);
    l.push(3);
    l.push(4);
    l.push(5);
    cout<<"Stack state after PUSH Operation:"<<endl;
    cout<<"================================"<<endl;
    l.display();
    cout<<endl;

    l.pop();
    cout<<"Stack state after POP Operation:"<<endl;
    cout<<"================================"<<endl;
    l.display();
    cout<<endl;

    cout<<"Stack Front Value:"<<endl;
    cout<<"==================="<<endl;
    cout<<l.getFront();
    cout<<endl;

    getch();
}
```

Output:



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:

```
#include <string>
using namespace std;

#pragma once
class student
{
public:
    int id;
    string name;
    int age;
    student *prev;
    student *next;
public:
    student(void);
};
```

Student.cpp File:

```
#include "student.h"
#include <string>
#include <iostream>
using namespace std;

student::student(void)
{
}
```

studentList.h File:

```
#include "student.h"
#include <string>
using namespace std;

#pragma once
class studentList
{
public:
    student *head;
public:
    studentList(void);
    bool isEmpty();
    void addStudent(int, string, int);
    void findStudent(string);
    void display();
};
```

studentList.cpp File:

```
#include "studentList.h"
#include "student.h"
#include <string>
#include <iostream>
using namespace std;

```

```
studentList::studentList(void)
{
    head = '\0';
}

bool studentList::isEmpty()
{
    if(head == '\0')
    {
        return true;
    }
    else
    {
        return false;
    }
}

void studentList::addStudent(int id, string name, int age)
{
    student *ptr = new student;
    ptr->id = 0;
    ptr->name = '\0';
    ptr->age = 0;
    ptr->prev = '\0';
    ptr->next = '\0';

    ptr->id = id;
    ptr->name = name;
    ptr->age = age;

    if(!isEmpty())
    {
        ptr->next = head;
        head->prev = ptr;
        head = ptr;
    }
    else
    {
        head=ptr;
    }
}

void studentList::findStudent(string name)
{
    if(!isEmpty())
    {
        student *temp;
        temp = head;

        while(temp->name != name && temp!= '\0')
        {
            temp = temp->next;
        }

        cout<<"Student Record:"<<endl;
        cout<<"===============<<endl;
        cout<<"Student ID: "<<temp->id<<endl;
        cout<<"Student Name: "<<temp->name<<endl;
        cout<<"Student Age: "<<temp->age<<endl;
    }
    else
    {
        cout<<"SORRY!!! Student List is Empty"<<endl;
    }
}

void studentList::display()
{
    if(!isEmpty())
    {
```

```
        student *temp;
        temp = head;

        cout<<"Student Record List:"<<endl;
        cout<<"===================="<<endl;
        while(temp != '\0')
        {
            cout<<"Student ID: "<<temp->id<<endl;
            cout<<"Student Name: "<<temp->name<<endl;
            cout<<"Student Age: "<<temp->age<<endl;
            cout<<endl;
            temp = temp->next;
        }
    }
    else
    {
        cout<<"SORRY!!! Student List is Empty"<<endl;
    }
}
```

Main.cpp File:

```
#include "studentList.h"
#include <string>
#include "student.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    studentList l;
    int choice;
    int id;
    string searchName;
    string name;
    int age;

    do
    {
        cout<<"Enter your desired operation:"<<endl;
        cout<<"1. ADD Student."<<endl;
        cout<<"2. SEARCH Student."<<endl;
        cout<<"3. DISPLAY Student List."<<endl;
        cin>>choice;
        cout<<endl;
        if(choice == 1)
        {
            cout<<"Enter Student ID:"<<endl;
            cin>>id;
            cout<<"Enter Student Name:"<<endl;
            cin.ignore(); //getline is having issue in while loop so that this statement is used
            getline(cin, name);
            cout<<"Enter Student Age:"<<endl;
            cin>>age;
            l.addStudent(id, name, age);
            cout<<endl;
        }
        else if(choice == 2)
        {
            cout<<"Enter search Student Name:"<<endl;
            cin>>searchName;
            cout<<endl;
            l.findStudent(searchName);
            cout<<endl;
        }
        else
        {
            l.display();
```

```
            cout<<endl;
        }
    }
    while(choice == 1 || choice == 2 || choice == 3); //1 for ADD STUDENT, 2 for SEARCH STUDENT, 3 for DISPLAY STUDENT LIST
    getch();
}
```

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.
1

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
```

10:14 PM
02-Apr-17