# Department of Computing

**CS250: Data Structure and Algorithms**

**Class: BSCS 9AB**

# Lab 2: Singly Linked Lists

# Course Instructor: Dr. Yasir Faheem

**Submitted By:**

**Fatima Seemab**

**CMS ID: 291310**

**BSCS 9-B**

**Lab # 2**

**CODE:**

#include <iostream>

using namespace std;

struct ListNode

{ int data;

ListNode\* next;

};

class LinkList

{

public:

ListNode\* start;

ListNode\* last;

ListNode\* temp;

ListNode\* loc;

ListNode\* ploc;

//Constructor of link list

//creates an empty list

LinkList()

{

start = nullptr;

last = nullptr;

}

//function return whether a list contains any element or not

bool isEmpty()

{return (start == nullptr);}

//Method to insert an element at front of list

void insertatFront(int value)

{

ListNode\* newnode = new ListNode();

//an object of listNode class iscreated

newnode->data = value;

//object of this data part is given value

if (isEmpty())

{

last = newnode;

start = newnode;

}

//if list is empty,After insertion of first element both start and last Pointers point to it.

else

{

newnode->data = value;

newnode->next = start;

start = newnode;

//start pointer is pointed to new node

}

}

void printLinkList()

{

//if list is empty,method will return

if (isEmpty())

{

cout << "nothing to show u";

return;

}

temp = start;

//condition valid tills the last of list

while (temp != last->next)

{

cout << temp->data;

//displays the element

temp = temp->next;

// temp pointer is incremented

}

cout << endl;

}

//Method to insert element at the end of list

void InsertAtend(int value)

{

ListNode\* newnode = new ListNode();

newnode->data = value;

//if List is empty,both start and last pointers ill be updates

if (isEmpty())

{

start = newnode;

last = newnode;

}

//if list contains any element,only last pointer will be updated

else

{

last->next = newnode;

last = newnode;

}

}

//Method to search a value

void Search(int value)

{//if list is empty,method will return

if (isEmpty())

{

cout << "There is no list to found";

return;

}

//Location and predecessor of location is initialized

loc = start;

ploc = NULL;

//Loop continues until list is finished or searched value will become greater then the currnt value of List

while((loc!=NULL) &&(loc->data<value))

{//Whenever a value is found,function will return

if (loc->data == value)

{return;

}

//stores the previous location of pointer

ploc = loc;

//Loc pointer is pointed to next

loc = loc->next;

}

//To make loc null when a value is not found

if ((loc!=NULL) && ((loc->data)!=value))

{ loc=NULL; }

}

// Method to insert an element at its sorted position

void InsertSorted(int value)

{// searcches the value in list

Search(value);

//when value is found in list

if (loc!=NULL)

{

cout << "value already exist in list";

}

//when value is in list and its logical position is at front

if ((loc==NULL) && (ploc == NULL))

{

insertatFront(value);

}

//If value is not in list and its position is not at front

if((loc==NULL) && (ploc!=NULL))

{

ListNode\* newnode = new ListNode();

//new object is created

newnode->data = value;

//pointed to appropriate position

newnode->next = ploc->next;

ploc->next = newnode;

//when elements logical position is at end

if (ploc == last)

{

last = newnode;

}

}

}

void Delete(int value)

{//searches a value

Search(value);

//If value is not found in list

if (loc == NULL)

{cout << "value is not present in list";

return;

}

else

{//value is present in list

if (ploc == NULL)

{//value is at start of list

start = loc->next;

delete loc;

}

//value is not at start of list

else

{

//value is at last of list

if (loc == last)

{

last = ploc;

}

//value is anywhere mid in list

else {

ploc->next = loc->next;

}

// deletes value

delete loc;

}

}

}

//Method to destroy list

void DestroyList()

{

temp = start;

//From start to last deletes all elements in list

while (temp != NULL)

{

start = start->next;

delete temp;

temp = start;

}

}

};

int main()

{

LinkList\* a=new LinkList();

cout << "An empty list is created"<<endl;

a->isEmpty();

a->insertatFront(7);

a->insertatFront(10);

a->InsertAtend(15);

cout << "List after insertion of elements at front and end"<<endl;

a->printLinkList();

cout << "List after insertion sorted of 3"<<endl;

a->InsertSorted(3);

a->printLinkList();

cout << "List after deletion of 10"<<endl;

a->Delete(10);

a->printLinkList();

cout << "List is destroyed" << endl;

a->DestroyList();

a->printLinkList();

}

**OUTPUT**

