Experiment No:5

Title: Implementation of Linked List Using C++.

Problem statement**:** Implementing linear data structure linked list using class and structure with functions

addNode()

delNode()

Display()

InsertNode()

SearchNode()

Algorithm:

**1.** Declare a structure Node with data and a pointer pointing to itself. Create a class with the member functions and head and tail pointers. In constructor of class initialize head and tail values as NULL.

**2.** Call the functions in a switch statement according to the choice value

**3 .** In addNode function declare a new node and ask for the value to be added as node. If there is no value in linked list point head and tail to the created node else point the tail to temp node and in tail->next pointed to NULL.

**4 .** In delNode function declare two new node pointers prev and curr and keep incrementing using curr pointer and if value at curr node is equal to one to be deleted make the prev->next equal to address of node one next to curr. By this we are skipping the node to be deleted.

**5.** In search function declare a pointer and keep incrementing until value at node matches and return the position

**6 .** In insert function take the value to be inserted in a temp node and the in temp->next give head address. By this we have inserted the node at beginning.

**7.** In display function declare a temp node with head as initial value and keep incrementing until temp becomes NULL and print value at node.

Code:

#include<iostream>

using namespace std;

struct Node{

int data;

Node \*next;

};

class linkedlist{

Node \*head,\*tail;

public :

linkedlist()

{

head = NULL;

tail = NULL;

}

void addnode(int n)

{

Node \*temp = new Node;

temp->data = n;

temp->next = NULL;

if(head == NULL)

{

head = temp;

tail = temp;

}

else{

tail->next = temp;

tail = tail->next;

}

}

void delnode()

{

int n;

Node \*prev = new Node;

Node \*curr = new Node;

prev = NULL;

curr = head;

cout<<"Enter the value to be deleted : ";

cin>>n;

while(curr != NULL)

{

if(curr->data == n)

{

prev->next = curr->next;

prev = prev->next;

return;

}

prev = curr;

curr = curr->next;

}

}

void searchnode()

{

int n,c=0;

Node \*temp = new Node;

temp = head;

cout<<"Enter the value to be searched :";

cin>>n;

while(temp != NULL)

{

if(temp->data == n)

{

cout<<"The element found at position "<<c<<endl;

}

temp = temp->next;

c = c + 1;

}

}

void insertnode()

{

Node \*temp = new Node;

int n;

cout<<"Enter the value :";

cin>>n;

temp->data = n;

temp->next = NULL;

temp->next = head;

head = temp;

}

void display()

{

Node \*temp = new Node;

temp = head;

while(temp!=NULL)

{

cout<<temp->data<<"\t";

temp = temp->next;

}

cout<<endl;

}

};

int main()

{

linkedlist L;

int ch,n;

start :

cout<<"Enter the choice\n1.Add\t2.Delete\t3.Search\t4.Insert\t5.Display\t";

cin>>ch;

switch(ch)

{

case 1 :

cout<<"Enter the value : ";

cin>>n;

L.addnode(n);

break;

case 2 :

L.delnode();

break;

case 3 :

L.searchnode();

break;

case 4 :

L.insertnode();

break;

case 5 :

L.display();

break;

default :

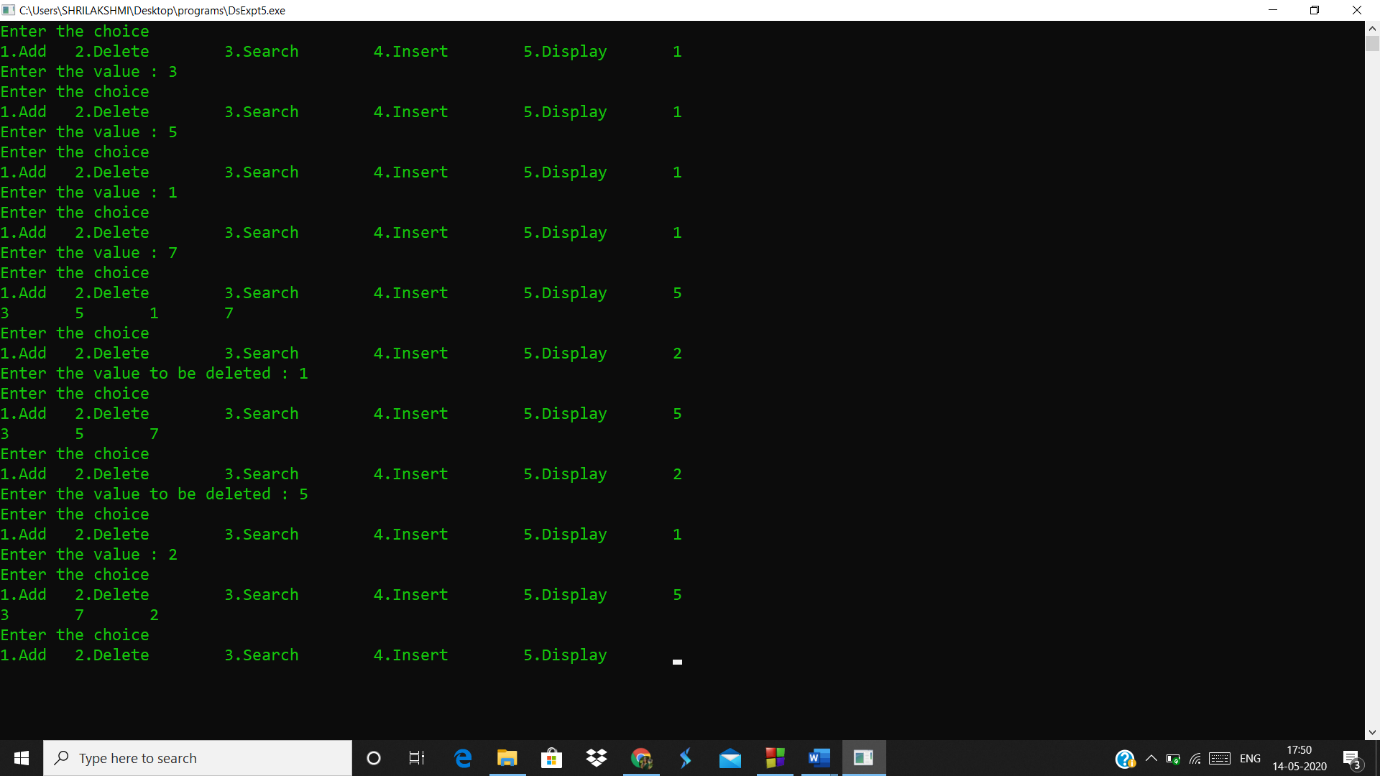
cout<<"Enter correct choice\n";

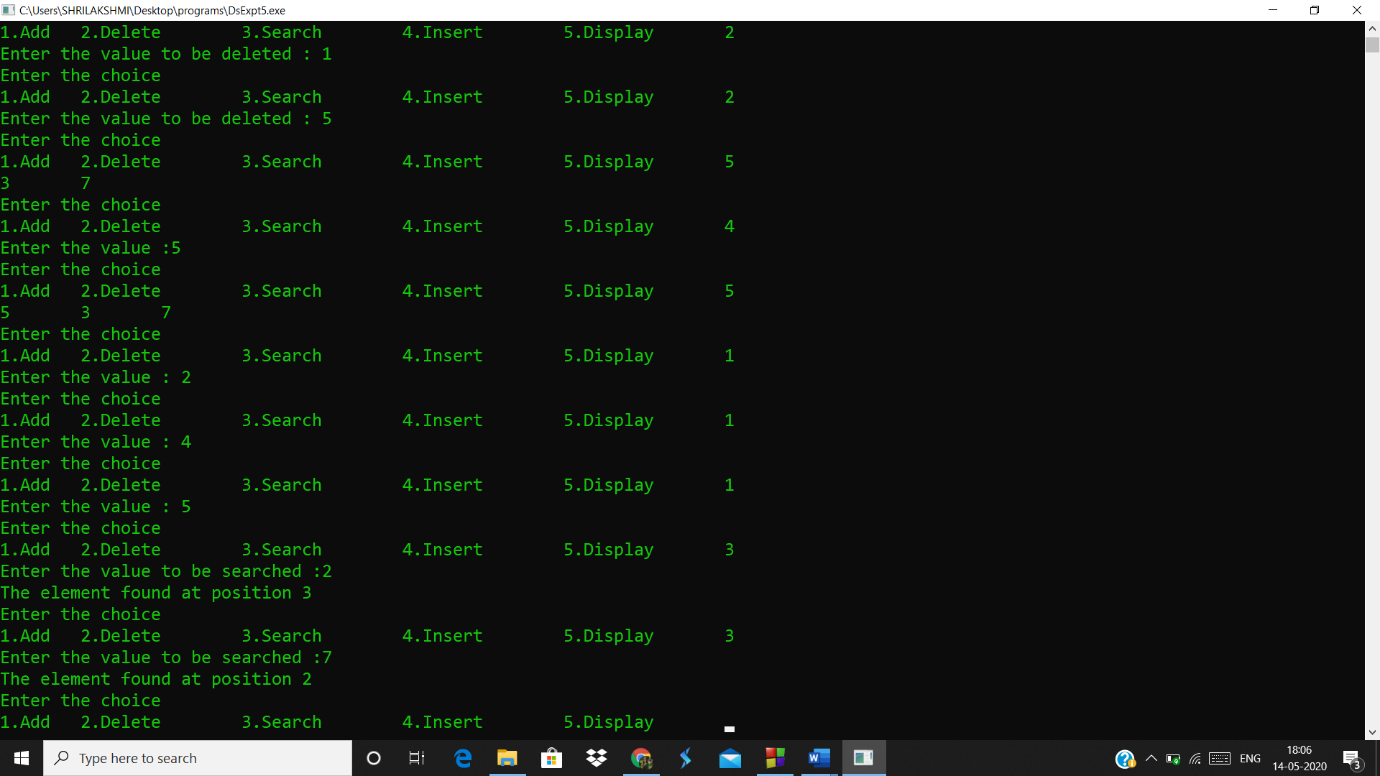
}

goto start;

}

Output:





Analysis(limitations):

* Value can be added only at the beginning, we cannot add inbetween.
* Since the program requires many nodes and address for each will be held by previous node any mislead in address may lead linked list to fail.