**Experiment Number: 5**

**Title: I**mplementation of Linked List Using C++.

**Problem Statement:** Write a menu driven C++ program to create the linked list using C++ and perform the following functions.

1. Create
2. Insert
3. Delete
4. Search
5. Display

**Algorithm:**

1. Start.
2. Create a class called Node. In Node class declare a variable called data to store the data of node and declare a pointer to the object of class Node called next to hold the address of the next node.
3. Create a child class of class Node called linkedlist.
4. In this class declare two pointers called head and tail which represent the first and the last nodes of the list.
5. Initialize these pointers to NULL using constructor function.
6. Declare the functions to create, insert, delete, search and display.
7. Display the operations to the user and ask them to enter the choice and based on the choice perform the operations.
8. I) For create operation,
9. Declare the pointer which points to the new object of class Node so that it points to new memory location.
10. Read the value from the user and store it in the data variable which is pointed by pointer created and make the next address field of node NULL so that it points to nothing.
11. Check if the head of the list is NULL means list is empty,

If yes, then store the address of the current element in the head and tail which become the first and the last nodes respectively.

If No, then store the address of the current element in the next address field of the tail so that the last element becomes the current element and store the address of the current element in the tail pointer which becomes the tail of the list.

II) For insert operation,

1. Declare the pointer which points to the new object of class Node so that it points to new memory location.
2. Read the value from the user and store it in the data variable which is pointed by pointer created and make the next address field of node NULL so that it points to nothing.
3. Assign the next address field of the current element with the address of the head of the list so that the current element becomes the head of the list and store address of current element in the head variable.

III) For delete operation,

1. First check if the list is empty or not.

If Yes, then display deletion not possible.

If No, then ask the user to enter the position of the element , search the position and then load the next address field of the previous node (position) with the next address field of the current node(position) so that the previous node pointer points to the element that is next to the current element.

IV) For Search operation,

1. First check if the list is empty or not.

If Yes, then display empty list.

If No,

1. Then create the pointer of object of the class Node and load the value of head into the pointer.
2. Ask the user to enter the value to be searched.
3. Compare the value with the data pointed by the pointer If they are same then display the position of the element and stop. Load the pointer with the next address field of the node to point to the next element. Repeat this step till the last node or element is found.
4. Display the message element not found.

V) For display operation,

i. Create the pointer of object of class Node and load the value of head into the pointer.

ii. Check if pointer is NULL if yes, display empty list

If No,

Starting from head till tail display the elements.

**Code:**

#include<iostream>

using namespace std;

class Node

{ public:

int data;

Node\* next;

};

class LinkedList:public Node

{

Node \*head,\*tail;

public:

LinkedList()

{

head=NULL;

tail=NULL;

}

void createNode();

void insert();

void delet();

void display();

void search();

};

int main()

{

LinkedList lst;

int ch;

while(1)

{

cout<<"\n\*\*\*\* MENU \*\*\*\*";

cout<<"\n1:CREATE\t2: INSERT\t3:DELETE\t4:SEARCH\t5: DISPLAY\t6:EXIT\n";

cout<<"\nEnter Your Choice:";

cin>>ch;

switch(ch)

{

case 1:lst.createNode();

break;

case 2:lst.insert();

break;

case 3:lst.delet();

break;

case 4:lst.search();

break;

case 5:lst.display();

break;

case 6:return 0;

}

} return 0;

}

void LinkedList::createNode()

{

Node \*temp=new Node;

int n;

cout<<"\n Enter an Element:";

cin>>n;

temp->data=n;

temp->next=NULL;

if(head==NULL)

{

head=temp;

tail=head;

}

else

{

tail->next=temp;

tail=temp;

}

}

void LinkedList :: insert()

{

int n;

Node \*temp=new Node;

cout<<"\nEnter an Element:";

cin>>n;

temp->data=n;

temp->next=NULL;

temp->next=head;

head=temp;

}

void LinkedList::delet()

{

Node \*prev=NULL,\*cur=head;

int count=1,pos,ch;

if(cur!=NULL)

{

cout<<"\nEnter the Position of Deletion:";

cin>>pos;

while(count!=pos)

{

prev=cur;

cur=cur->next;

count++;

}

if(count==pos)

{

cout<<"\nDeleted Element is: " <<cur->data;

prev->next=cur-> next;

}

}

else

cout<<"\nNot Able to Delete";

}

void LinkedList::display()

{

Node \*temp=head;

if(temp==NULL)

{

cout<<"\nList is Empty";

}

while(temp!=NULL)

{

cout<<temp->data;

cout<<"-->";

temp=temp->next;

}

cout<<"NULL";

}

void LinkedList::search()

{

int value,pos=0;

bool flag=false;

if(head==NULL)

{

cout<<"\nList Empty";

return;

}

cout<<"\nEnter the number to be searched:";

cin>>value;

Node \*temp;

temp=head;

while(temp!=NULL)

{

pos++;

if(temp->data==value)

{

flag=true;

cout<<"Element"<<value<<"found at position"<<pos;

return;

}

temp=temp->next;

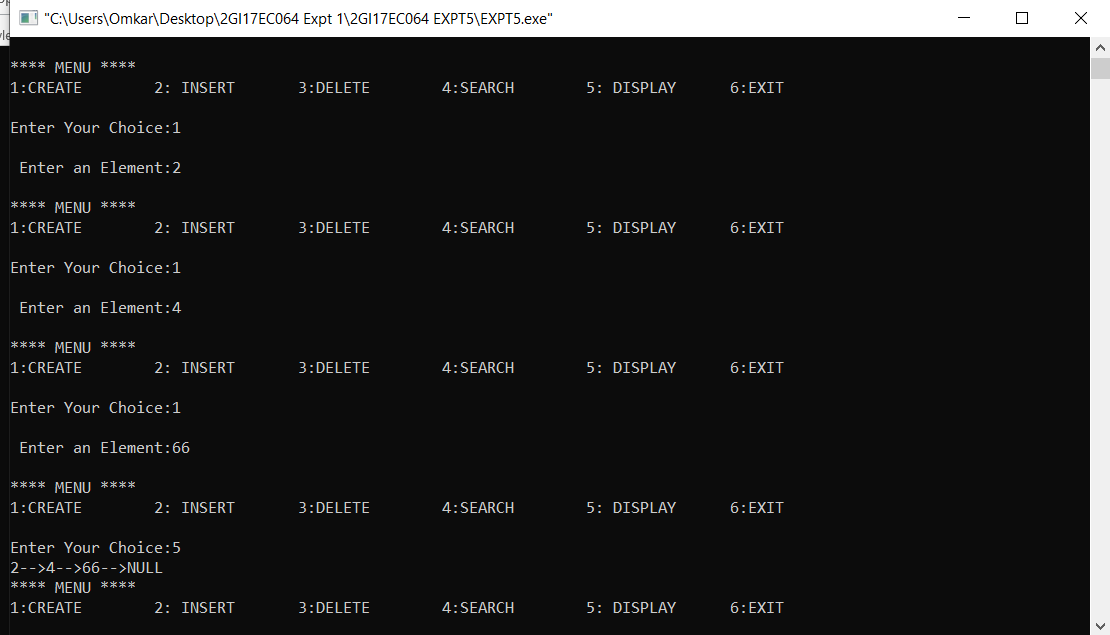
}

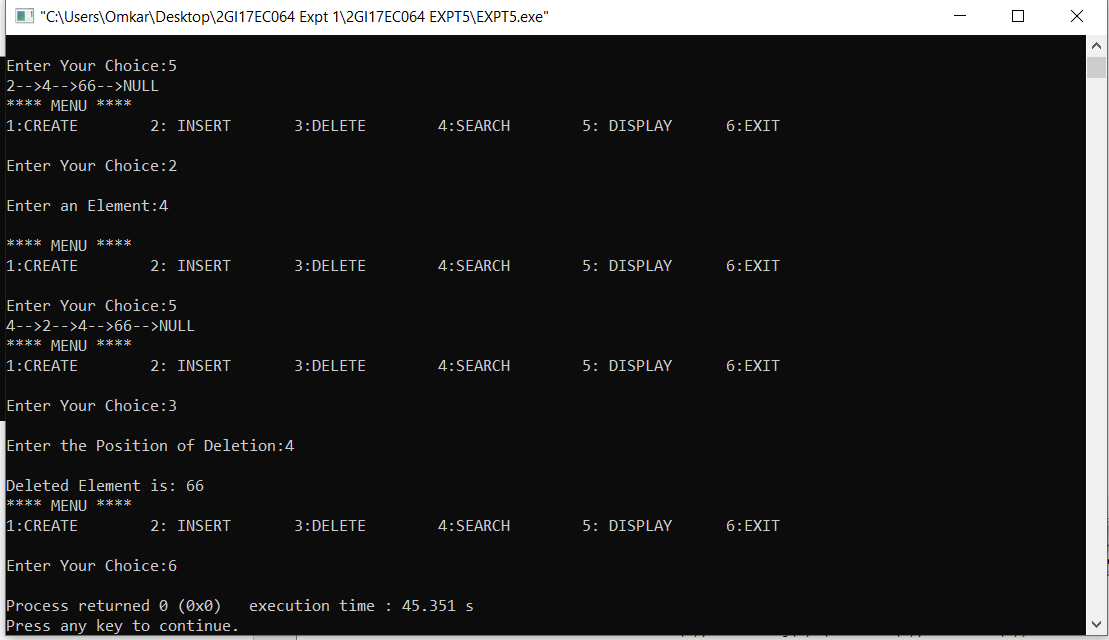
if(!flag)

cout<<"\nElement not found";

}

**Output:**





**ANALYSIS(LIMITATIONS):**

More memory is required to store elements **in linked list** as compared to array. Because **in linked list** each node contains a pointer and it requires extra memory **for** itself.

Elements or nodes traversal is difficult **in linked list**