**Experiment No. 5**

**Title :** Implementation of Linked list Linear Data Structure

**Problem Statement :**Write a C++ program to implement linked list data structure which includes following functionalities addnode() ,delnode() ,display() ,insertnode() ,searchnode()

**Algorithm :**

**Step 1**: Start

**Step 2:** Declare a structure node which has integer data variable and a structure pointer of node named next to hold address of nest node.

**Step 3:** Create class linkedlist with constructor having head and tail equal to NULL which are instances of structure node.

**Step 4:** Declare function addnode() to add node in this check if head is pointing to NULL, if yes then make the new node as head else append the new node after tail and update the tail address.

**Step 5:** Declare delnode() function to delete node in which traverse through the linked list and compare the data field with each node data field and if fount then skip the node by changing the next field of previous node pointing to next node.

**Step 6:** Declare searchnode() function in which traverse through the linked list by comparing the data field of each node with the desired node’s data field and keep count of index ,if data field matches then return the index of the node in linked list else display not found.

**Step 7 :** Declare insert() which inserts value at the head position of the linked list and display() function to display the linked list by traveling from head to NULL address.

**Step 8:** Input choice from user in infinite loop for above functionalities and switch according to the choice entered to the function.

**Step 9:** Stop

**Program:**

#include<iostream>

using namespace std;

struct Node{ //Declare the structure Node

int data;

Node \*next; //pointer pointing to Node structure

};

class linkedlist{ //Declare class linked list

Node \*head,\*tail;

public :

linkedlist() //Constructor to make head and tail to NULL initially

{

head = NULL;

tail = NULL;

}

void addnode(int n) // Declare add node function

{

Node \*temp = new Node; //Create a new node by structure Node

temp->data = n;//store data

temp->next = NULL;//make next location address as NULL

if(head == NULL)//if no node exist make the node as head node

{

head = temp;

tail = temp;

}

else{//append the new node at the end of linked list

tail->next = temp;//point tail to new node

tail = tail->next;//update tail to new nodes next position

}

}

void delnode() //Delete Node

{

int n;

Node \*prev = new Node; //Previous node tracer

Node \*curr = new Node; //Current node under value comparision

prev = NULL;

curr = head;

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

cin>>n;

while(curr != NULL)//traverse through the linked list

{

if(curr->data == n)//compare the entered value with data part of each element of linked list

{

prev->next = curr->next; //Skip the address of node to be deleted by giving address of next node

prev = prev->next;

return;

}

prev = curr;//previous will become current

curr = curr->next;//current will point to next node

}

}

void searchnode() //Search a node in linked list by value

{

int n,c=0;

Node \*temp = new Node;//for traversing through linked list

temp = head;//points to head

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

cin>>n;//input the value

while(temp != NULL)//traverse till end of linked list

{

if(temp->data == n)//compare the entered value with data part of each element of linked list

{

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

}

temp = temp->next; //point to node for traversal

c = c + 1;//position increment

}

}

void insertnode() //Insert node inserts node at the beginning of linked list

{

Node \*temp = new Node;//temporary node which will be made as head node

int n;//to take data value

cout<<"Enter the value :";

cin>>n;//input data value

temp->data = n;//put n in data part of temp node

temp->next = NULL;//point next to NULL address

temp->next = head;//now point next to head address

head = temp;//make temp as head

}

void display() //Display function

{

cout<<"\nLinked lit is : \n";

Node \*temp = new Node;//for traversal through linked list

temp = head;//point temp to head node

while(temp!=NULL)//till end of linked list

{

cout<<temp->data<<" --> ";//display data part of each node

temp = temp->next;////point temp to next node

}

cout<<endl;

}

};

int main()

{

cout<<"\*\*\*\* L I N K E D L I S T \*\*\*\*"<<endl;

linkedlist L; //Instantiate L as object of linked list class

int choice,n;

while(1)

{

cout<<"\n 1\_Add\t2\_Delete\t3\_Search\t4\_Insert\t5\_Display\t6\_Exit\nEnter the choice:";

cin>>choice;//input choice

switch(choice)

{

case 1 ://add node

cout<<"Enter the value : ";

cin>>n;//input data part of node being added

L.addnode(n);

break;

case 2 ://delete node

L.delnode();

break;

case 3 ://search for node location depending on data stored

L.searchnode();

break;

case 4 ://insert node

L.insertnode();

break;

case 5 ://display linked list

L.display();

break;

case 6:return(0);

default :

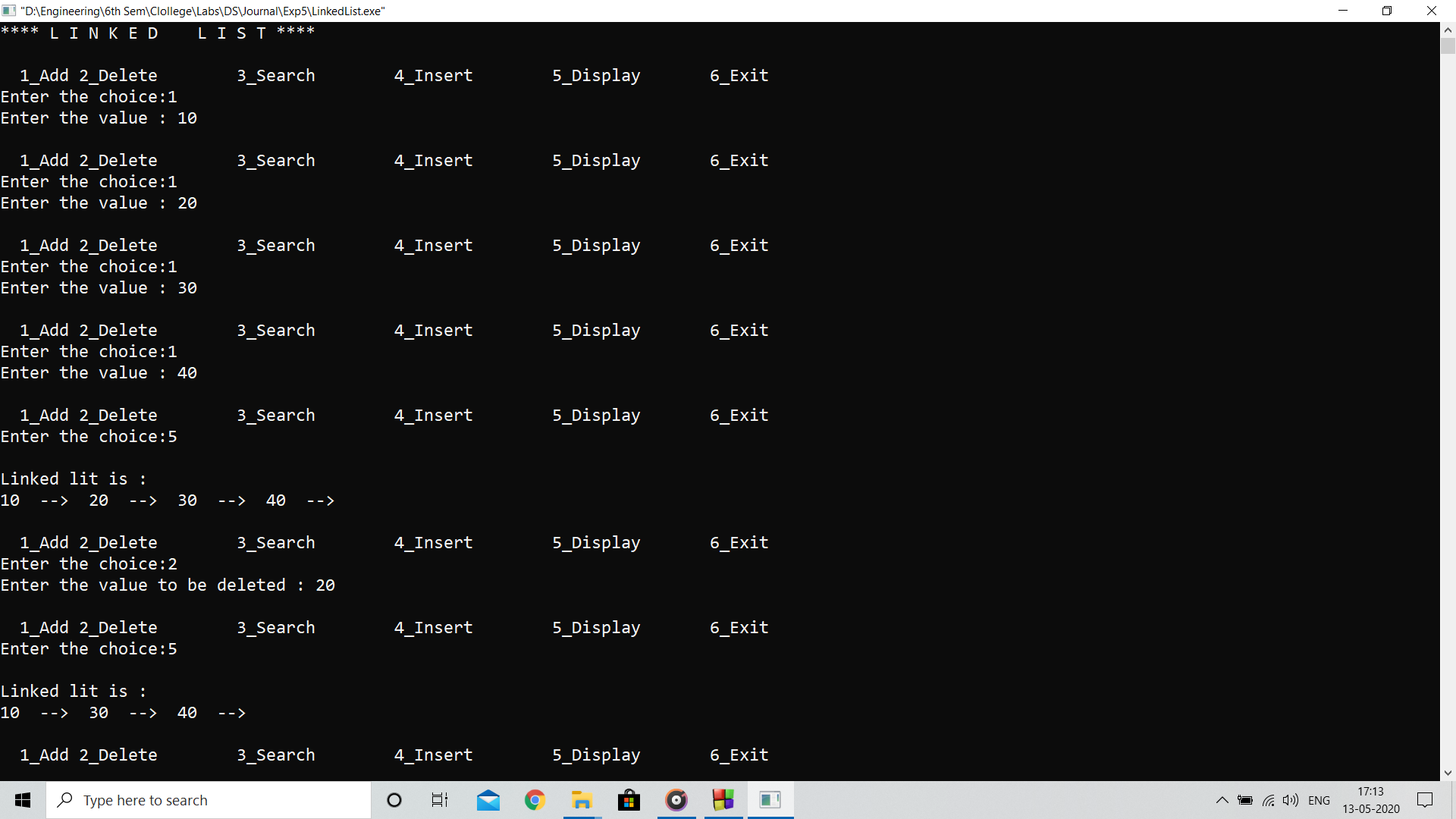
cout<<"Enter correct choice\n";

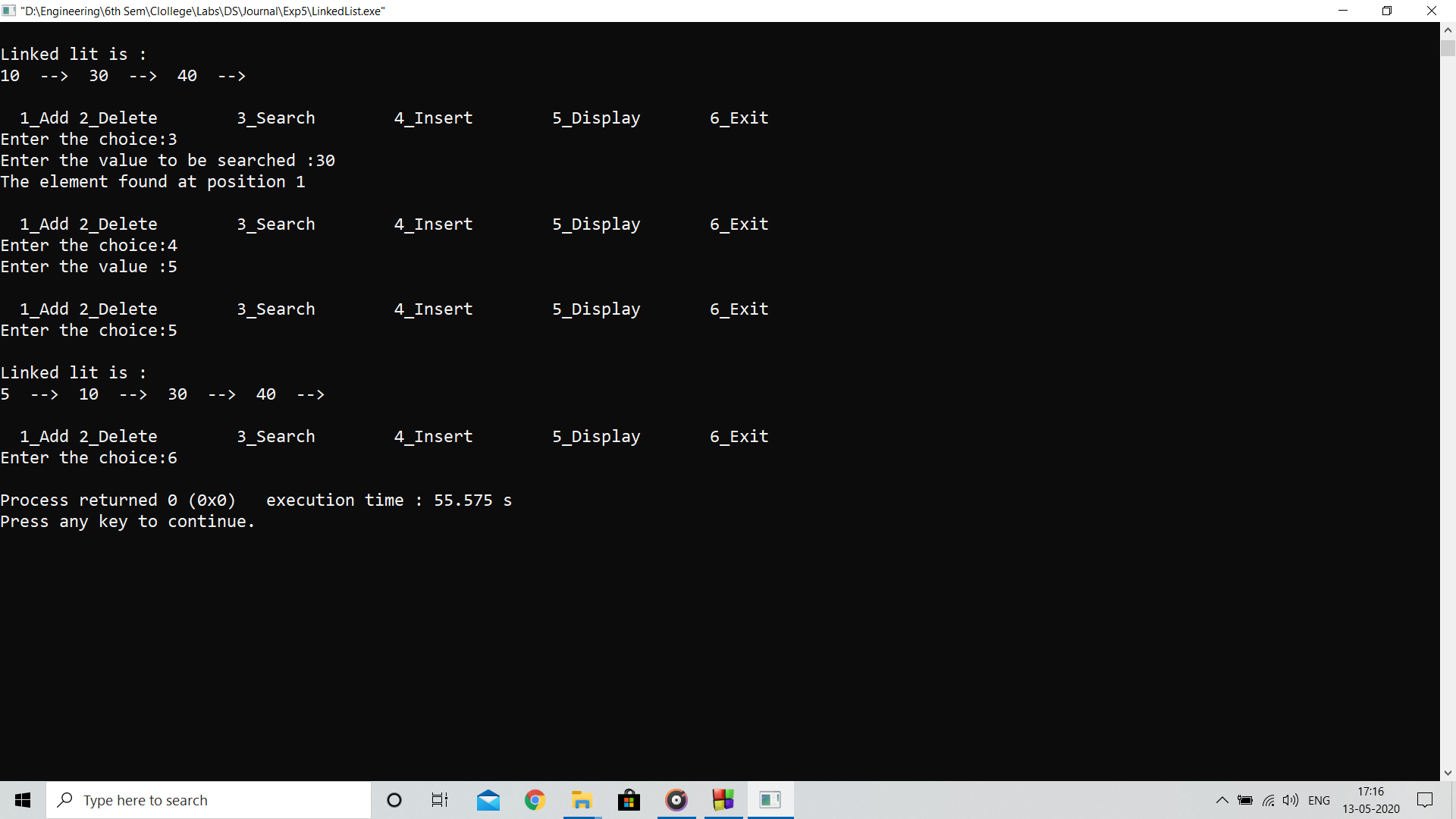
}//end switch

}//end while

}//end main

**Output :**

****

****

**Analysis:**

Program implements the linked list data structure with all functionalities of add, delete, insert and search of a node in linked list

**Limitation:**

Insert function only inserts the node at the head position in linked list although it should insert at desired location