Practical-6

AIM: Implement below operations of singly linked list.

(a) Insert a node at front

(b) Delete a node at last

(c) Delete all nodes of linked list

Note: Display content of linked list after each operation.

* Program

#include<bits/stdc++.h>

using namespace std;

struct node{

int info;

struct node \* link;

};

struct node\* First = NULL;

struct node\* firstNode = NULL;

void pushAtFirst(int x);

void pushAtEnd(int x);

void pushAtRan(int a, int b);

struct node\* Create(int n);

void deleteLast();

void deleteFirst();

void deleteAll();

void display();

int main(){

int choice;

cout << "This program is developed by 22CE097\_ShivangPatel" << endl << endl;

do{

cout << "1. Insert node at front." << endl;

cout << "2. Inset node at end." << endl;

cout << "3. Inset node before specific value." << endl;

cout << "4. Delete node at last." << endl;

cout << "5. Delete node at front." << endl;

cout << "6. Delete all the nodes." << endl;

cout << "7. Display" << endl;

cout << "8. Exit" << endl;

cout << "Select operation : ";

cin >> choice;

switch (choice)

{

case 1:

int x;

cout << "Enter a element you want to add : ";

cin >> x;

pushAtFirst(x);

display();

break;

case 2:

int y;

cout << "Enter a element you want to add : ";

cin >> y;

pushAtEnd(y);

display();

break;

case 3:

int a, b;

cout << "Enter a element you want to add : ";

cin >> a;

cout << "Enter a element before you want to insert " << a << " : ";

cin >> b;

pushAtRan(a, b);

display();

break;

case 4:

deleteLast();

display();

break;

case 5:

deleteFirst();

display();

break;

case 6:

deleteAll();

display();

break;

case 7:

display();

break;

case 8:

break;

default:

cout << "Select a valid number" << endl;

break;

}

}while(choice!=8);

return 0;

}

struct node\* Create(int x){

struct node\* temp;

temp = (struct node\*)malloc(sizeof(struct node\*));

temp->info = x;

temp->link = NULL;

return temp;

}

void pushAtFirst(int x){

struct node\* temp = Create(x);

if(First == NULL){

First = temp;

//firstNode = First->link;

}

else{

temp->link = First;

First = temp;

}

}

void pushAtEnd(int x){

struct node\* temp = Create(x);

if(First == NULL){

First = temp;

}

else{

struct node\* trav = First;

while(trav->link!=NULL){

trav = trav->link;

}

trav->link = temp;

}

}

void pushAtRan(int a, int b)

{

struct node\* temp = Create(a);

struct node\* temp1 = First;

struct node\* pre = temp;

while(temp1->info!=b){

if(temp1->link==NULL){

cout << a << " is not found in stack." << endl;

return;

}

else{

pre = temp1;

temp1 = temp1->link;

}

}

temp->link = temp1;

pre->link = temp;

}

void deleteLast(){

struct node\* temp = First;

struct node\* pre = temp;

if(First == NULL){

cout << "Stack is underflow" << endl;

}

else if(temp->link == NULL){

free(temp);

First=NULL;

}

else{

while(temp->link!=NULL){

pre = temp;

temp = temp->link;

}

pre->link = NULL;

free(temp);

}

}

void deleteFirst(){

if(First == NULL){

cout << "Stack is underflow" << endl;

}

else{

struct node\* temp = First;

First = First->link;

free(temp);

}

}

void deleteAll()

{

struct node\* temp = First;

while(temp->link!=NULL){

temp=temp->link;

free(First);

First=temp;

}

First=NULL;

}

void display(){

struct node\* temp1 = First;

cout << endl << "Elemets : ";

while(temp1!=NULL){

cout << temp1->info << " ";

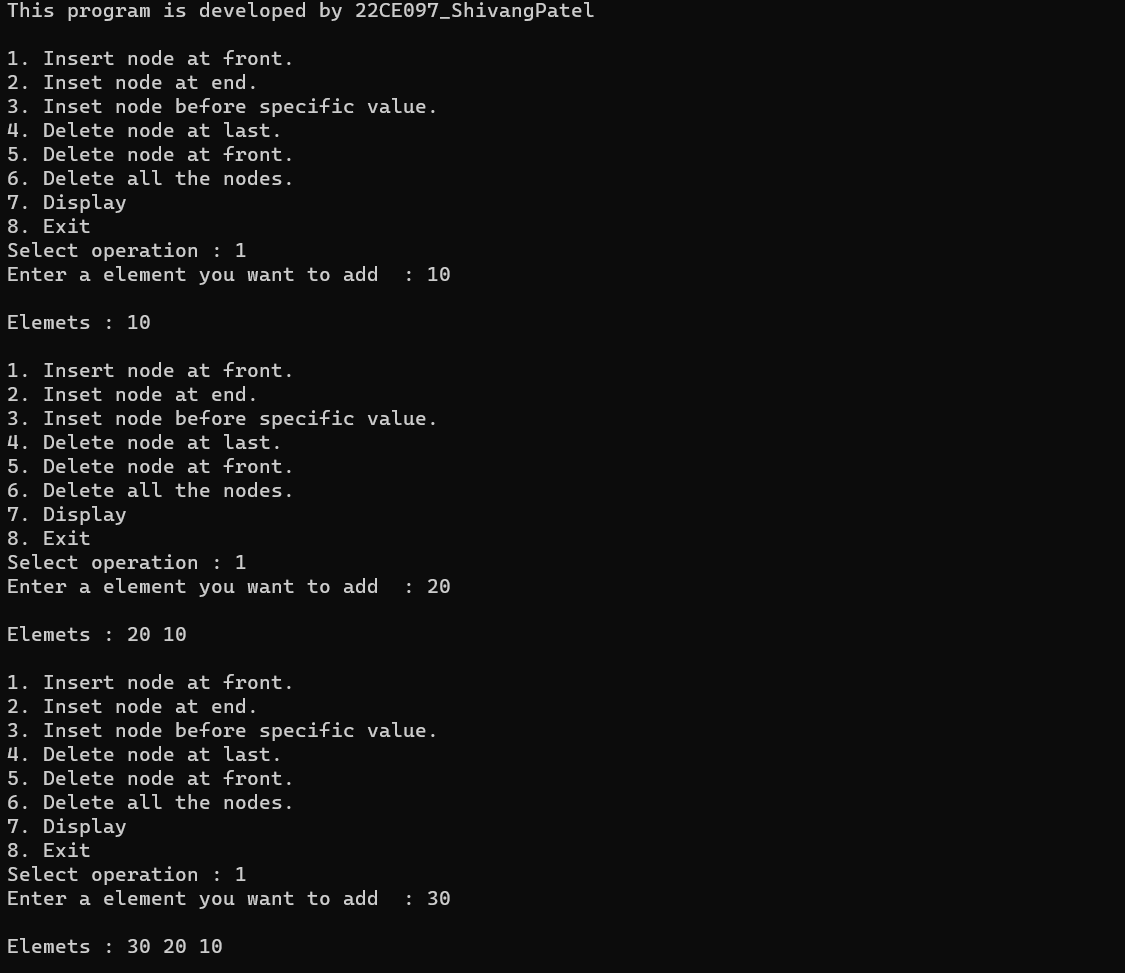
temp1=temp1->link;

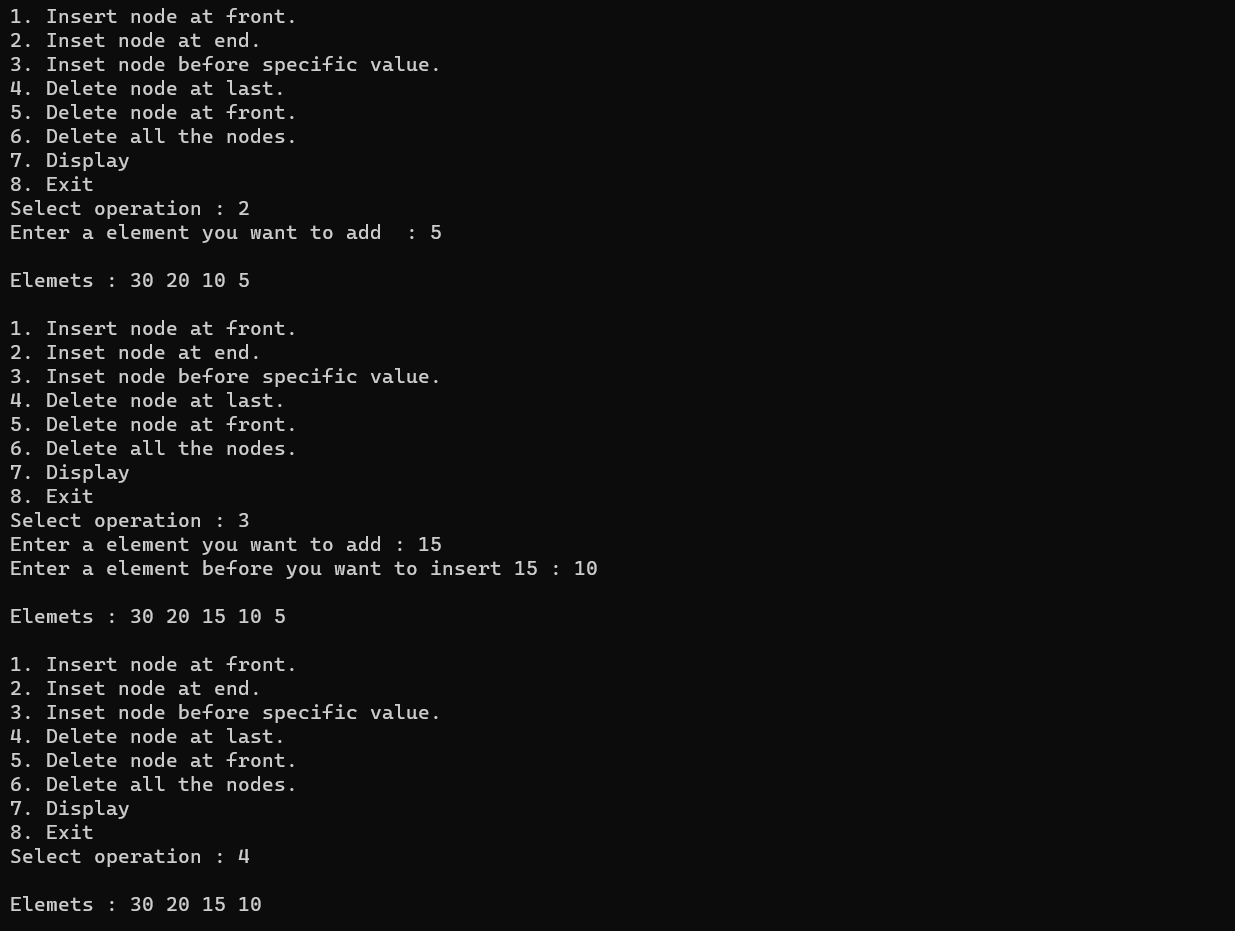
}

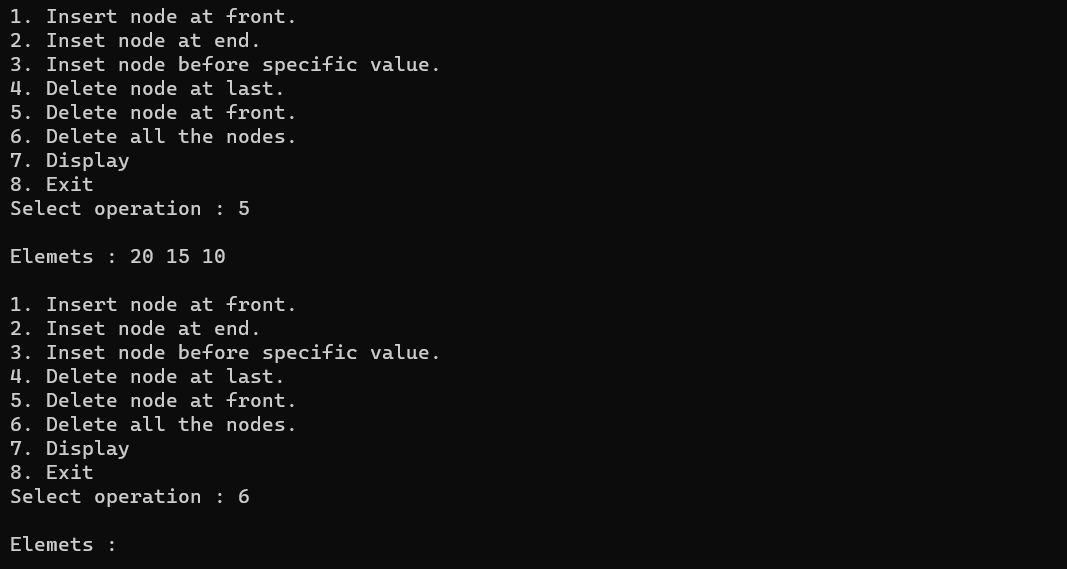
cout << endl << endl;

}

* Output







\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Student Signature Faculty Signature Marks