**Assignment No: 3**

**Problem Statement:**

Implement any database using a doubly-linked list with the following options

a) Insert a record

b) delete a record

c) modify a record

d) Display list forward

e) Display list backward

**Outcomes:**

Students will be able to:

1. Construct Linear data structures using arrays and linked lists.

**Code:**

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*next, \*prev;

};

class DoublyLinkedList{

Node \*head;

public:

DoublyLinkedList(){

head = NULL;

cout<<"Doubly Linked List is created successfully !"<<endl;

}

void insertElement(){

Node \*newNode = new Node;

newNode -> next = NULL;

newNode -> prev = NULL;

cout<<"\nEnter the data in the node: "<<endl;

cin>>newNode -> data;

if(! head){

head = newNode;

}

else{

Node \*temp = head;

while(temp -> next != NULL){

temp = temp -> next;

}

temp -> next = newNode;

newNode -> prev = temp;

}

cout<<newNode -> data<<" is inserted !"<<endl;

}

void deleteElement(){

int num;

if(!head){

cout<<"The list is empty !";

return;

}

cout<<"\nEnter the element to delete: ";

cin>>num;

Node \*temp = head;

while(temp != NULL && temp->data != num){

temp = temp -> next;

}

if(temp == head){

temp -> next ->prev = NULL;

head = temp -> next;

delete temp;

}

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

temp -> prev ->next = NULL;

delete temp;

}

else if(temp == NULL){

cout<<endl<<num<<" is not found in the list !";

return;

}

else{

temp -> prev -> next = temp -> next;

temp -> next -> prev = temp -> prev;

delete temp;

}

cout<<endl<<num<<" is deleted from the list !";

return;

}

void modify(){

if(!head){

cout<<"\nThe list is empty !"<<endl;

return;

}

int key;

cout<<"Enter the number which you want to modify: ";

cin>>key;

Node \*temp = head;

while(temp && temp -> data != key){

temp = temp -> next;

}

if(!temp){

cout<<endl<<key<<" is not found in the list !";

}

else{

int num;

cout<<"Enter the new number: ";

cin>>temp->data;

cout<<"Modified successfully !";

}

}

void display(){

if(!head){

cout<<"The list is empty !";

return;

}

Node \*temp = head;

cout<<"\n\nDoubly Linked List Data: "<<endl;

while(temp){

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

temp = temp -> next;

}

}

void displayReverse(){

if(!head){

cout<<"The list is empty !";

return;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

cout<<"\n\nDoubly Linked List Data in Reversed Order: "<<endl;

while(temp){

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

temp = temp -> prev;

}

}

~DoublyLinkedList(){

Node \*temp = head;

Node \*tempSec = temp;

while(temp){

tempSec = temp;

temp = temp -> next;

delete tempSec;

}

}

};

int main(){

DoublyLinkedList list;

int choice;

do{

cout<<"\n#############-MENU-##################\n";

cout<<"\n1. Insert \n2. Delete\n3. Modify\n4. Display\n5. Display Reverse\n6. Exit\n\n";

cout<<"\nEnter your choice: ";

cin>>choice;

switch(choice){

case 1: list.insertElement(); break;

case 2: list.deleteElement(); break;

case 3: list.modify(); break;

case 4: list.display(); break;

case 5: list.displayReverse(); break;

case 6: cout<<"\nShutting down..."; break;

default: cout<<"\nInvalid Choice !";

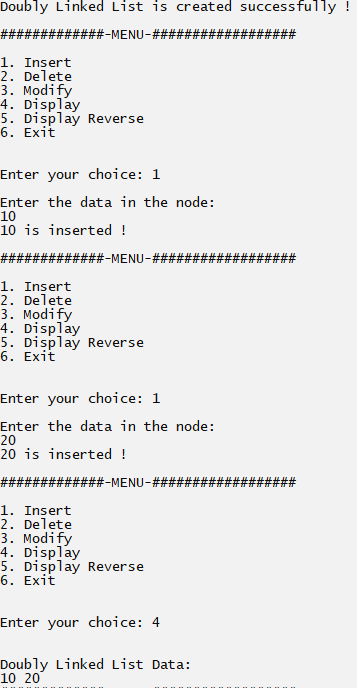
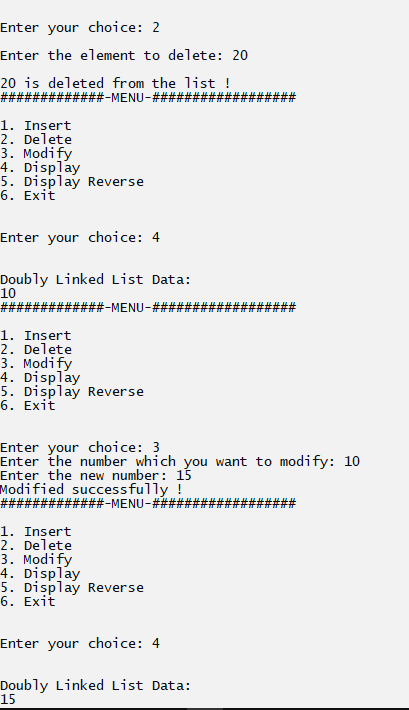
}

}while(choice!=6);

return 0;

}

**Output:**

1) 2)