**W1 LAB-B**

Q1))

1. Given a linked list, write functions to

a. Insert an element at the beginning of the linked list.

b. Insert an element at a specific location in a linked list.

c. Take input an integer number, split the number in its digits and stores the

digits in a linked list structure.

Sol))

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node \*next;

};

LinkedListTraversal(Node \*head){

Node \*ptr=head;

while(ptr!=NULL){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

Node \*InsertAtHead(Node \*head,int data){

struct Node \*ptr=new Node();

ptr->data=data;

ptr->next=head;

return ptr;

}

Node \*InsertAtIndex(Node \*head,int data,int index){

struct Node \*ptr=head;

struct Node \*p=new Node();

int i=0;

while(i!=index-1){

ptr=ptr->next;

i++;

}

p->data=data;

p->next=ptr->next;

ptr->next=p;

return head;

}

Node \*InsertAtEnd(Node \*head,int data){

struct Node \*ptr=head;

ptr->data=data;

struct Node \*p=new Node();

p->next=NULL;

p->data=data;

while(ptr->next!=NULL){

ptr=ptr->next;

}

ptr->next=p;

return head;

}

Node \*InsertDigits(Node \*head, int data){

int digit=0;

while(data>0){

digit=data%10;

head=InsertAtEnd(head,digit);

data/=10;

}

return head;

}

int main(){

struct Node \*head=NULL;

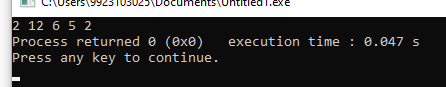
head=InsertAtHead(head,11);

head=InsertAtIndex(head,12,1);

head=InsertDigits(head,256);

LinkedListTraversal(head);

}



Q2)) Write a program which reads a name and generates the link list of the characters

in that name. Later it removes the vowels from the link list and displays the

modified link list.

Sol))

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

struct Node{

char data;

Node \*next;

};

void LinkedListTraversal(Node \*head){

Node \*ptr=head;

while(ptr!=NULL){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

Node \*InsertAtEnd(Node \*head,char data){

struct Node \*ptr=head;

struct Node \*p=new Node();

p->data=data;

p->next=NULL;

if(head==NULL){

return p;

}

while(ptr->next!=NULL){

ptr=ptr->next;

}

ptr->next=p;

return head;

}

Node \*InsertCharacters(Node \*head, string data){

char ch;

for(int i=0;i<data.length();i++){

ch=data.at(i);

head=InsertAtEnd(head,ch);

}

return head;

}

Node \*Remove(Node \*head, char ch){

if(head==NULL){

return head;

}

if(head->data==ch){

struct Node\*temp=head;

head=head->next;

delete temp;

}

struct Node\*ptr=head;

while(ptr->next!=NULL){

if(ptr->next->data==ch){

Node\*temp=ptr->next;

ptr->next=ptr->next->next;

delete temp;

}

else{

ptr=ptr->next;

}

}

return head;

}

Node \* RemoveVowels(Node \*head,string data){

head= Remove(head,'a');

head= Remove(head,'e');

head= Remove(head,'i');

head= Remove(head,'o');

head= Remove(head,'u');

head= Remove(head,'A');

head= Remove(head,'E');

head= Remove(head,'I');

head= Remove(head,'O');

head= Remove(head,'U');

return head;

}

int main(){

string s;

cout<<"enter the string: ";

getline(cin,s);

struct Node \*head=NULL;

head=InsertCharacters(head,s);

LinkedListTraversal(head);

head=RemoveVowels(head,s);

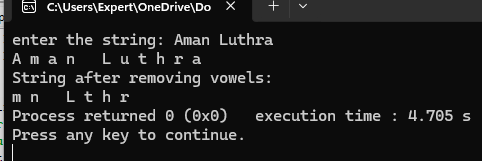
cout<<endl;

cout<<"String after removing vowels:";

cout<<endl;

LinkedListTraversal(head);

}



Q3))

Create a link list of users supplied ten characters to store a name. Create a

second link list of same type of user supplied five characters. Now using a

function remove(), traverse first link list and if any three consecutive characters

of second link list appears as consecutive characters of first link list, remove

those from first link list.

Sol))

#include<iostream>

#include<bits/stdc++.h>

#include<string>

using namespace std;

struct Node{

char data;

Node \*next;

};

void LinkedListTraversal(Node \*head){

Node \*ptr=head;

while(ptr!=NULL){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

Node \*InsertAtEnd(Node \*head,char data){

struct Node \*ptr=head;

struct Node \*p=new Node();

p->data=data;

p->next=NULL;

if(head==NULL){

return p;

}

while(ptr->next!=NULL){

ptr=ptr->next;

}

ptr->next=p;

return head;

}

Node \*InsertCharacters(Node \*head, string data){

char ch;

for(int i=0;i<data.length();i++){

ch=data.at(i);

head=InsertAtEnd(head,ch);

}

return head;

}

Node \*RemoveConsecutive(Node \*head, string s1,int m){

int s=0;

while(s!=3){

struct Node\* ptr=head;

int i=0;

while(i!=m-1&&ptr!=NULL){

ptr=ptr->next;

i++;

}

struct Node\*temp=ptr->next;

ptr->next=ptr->next->next;

delete temp;

s++;

}

return head;

}

Node \*Remove(Node \*head,string s1,string s2){

int m=0;

for(int i=0;i<s1.length()-2;i++){

m++;

string substr1=s1.substr(i,3);

for(int j=0;j<s2.length()-2;j++){

string substr2=s2.substr(j,3);

if(substr1==substr2){

head=RemoveConsecutive(head,s1,m-1);

}

}

}

return head;

}

int main(){

string s1;

cout<<"enter the string1 of 10 characters: ";

getline(cin,s1);

string s2;

cout<<"enter the string2 of 5 characters: ";

getline(cin,s2);

struct Node \*head1=NULL;

head1=InsertCharacters(head1,s1);

struct Node \*head2=NULL;

head2=InsertCharacters(head2,s2);

cout<<endl;

cout<<"String before removing consecutive characters:";

LinkedListTraversal(head1);

cout<<endl;

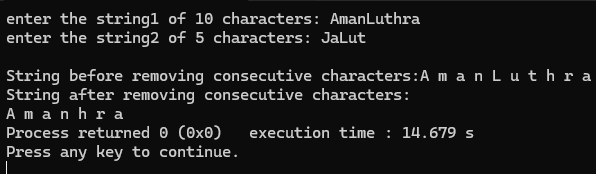
head1=Remove(head1,s1,s2);

cout<<"String after removing consecutive characters:";

cout<<endl;

LinkedListTraversal(head1);

}



Q4)) Write a program to insert an element at specific location in doubly linked list.

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*previous;

Node \*next;

Node(int val){

previous=NULL;

data=val;

next=NULL;

}

};

void InsertionAtHead(Node \* &head,Node \* &tail,int data){

Node \*ptr=new Node(data);

ptr->next=head;

if(head!=NULL){

head->previous=ptr;

}

else{

tail=ptr;

}

head=ptr;

}

void InsertAtTail(Node \* &head, Node \* &tail,int data){

Node \*ptr=new Node(data);

if(tail!=NULL){

tail->next=ptr;

ptr->previous=tail;

}

else{

head=ptr;

}

tail=ptr;

}

void TraversalFromHead(Node \* &head){

Node \*temp=head;

while(temp!=NULL){

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

temp=temp->next;

}

cout<<endl;

}

void TraversalFromTail(Node \* &tail){

Node \*temp=tail;

while(temp!=NULL){

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

temp=temp->previous;

}

cout<<endl;

}

void InsertAtIndex(Node \* &head, Node \* &tail,int data,int index){

if(index==0){

InsertionAtHead(head,tail,data);

return;

}

Node \*ptr=new Node(data);

Node \*temp=head;

int i=0;

while(i!=index-1){

temp=temp->next;

i++;}

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

InsertAtTail(head,tail,data);

return;

}

ptr->next=temp->next;

temp->next=ptr;

ptr->previous=temp;

temp->next->previous=ptr;

}

int main(){

Node \*head=NULL;

Node \*tail=NULL;

InsertionAtHead(head,tail,0);

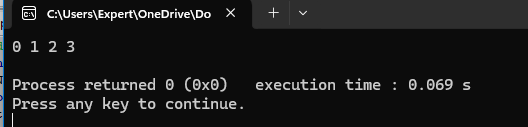
InsertAtIndex(head,tail,1,1);

InsertAtTail(head,tail,3);

InsertAtIndex(head,tail,2,2);

TraversalFromHead(head);

}



Q5)) Write a program to delete last element from the doubly linked list.

Sol))

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*previous;

Node \*next;

Node(int val){

previous=NULL;

data=val;

next=NULL;

}

};

void InsertionAtHead(Node \* &head,Node \* &tail,int data){

Node \*ptr=new Node(data);

ptr->next=head;

if(head!=NULL){

head->previous=ptr;

}

else{

tail=ptr;

}

head=ptr;

}

void InsertAtTail(Node \* &head, Node \* &tail,int data){

Node \*ptr=new Node(data);

if(tail!=NULL){

tail->next=ptr;

ptr->previous=tail;

}

else{

head=ptr;

}

tail=ptr;

}

void TraversalFromHead(Node \* &head){

Node \*temp=head;

while(temp!=NULL){

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

temp=temp->next;

}

cout<<endl;

}

void TraversalFromTail(Node \* &tail){

Node \*temp=tail;

while(temp!=NULL){

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

temp=temp->previous;

}

cout<<endl;

}

void InsertAtIndex(Node \* &head, Node \* &tail,int data,int index){

if(index==0){

InsertionAtHead(head,tail,data);

return;

}

Node \*ptr=new Node(data);

Node \*temp=head;

int i=0;

while(i!=index-1&&temp!=NULL){

temp=temp->next;

i++;}

if(temp->next==NULL){

InsertAtTail(head,tail,data);

return;

}

ptr->next=temp->next;

temp->next=ptr;

ptr->previous=temp;

if(ptr->next!=NULL)

ptr->next->previous=ptr;

}

void Deletelast(Node \* &head,Node \*&tail){

Node \*temp=tail;

if(tail==NULL){

return;

}

if(head==tail){

head=NULL;

tail=NULL;

}

else{

tail=tail->previous;

tail->next=NULL;}

delete temp;

}

int main(){

Node \*head=NULL;

Node \*tail=NULL;

InsertionAtHead(head,tail,0);

InsertAtIndex(head,tail,1,1);

InsertAtTail(head,tail,3);

InsertAtIndex(head,tail,2,2);

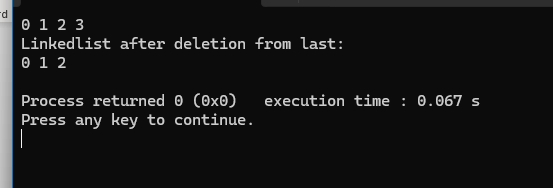
TraversalFromHead(head);

Deletelast(head,tail);

cout<<"Linkedlist after deletion from last:\n";

TraversalFromHead(head);

}



Q6)) Given a doubly linked list of any number of nodes, write a function

ExtremeSwap(), which will swap values of the node at extreme pairs.

Sol))

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node \*previous;

Node \*next;

Node(int val){

previous=NULL;

data=val;

next=NULL;

}

};

void InsertionAtHead(Node \* &head,Node \* &tail,int data){

Node \*ptr=new Node(data);

ptr->next=head;

if(head!=NULL){

head->previous=ptr;

}

else{

tail=ptr;

}

head=ptr;

}

void InsertAtTail(Node \* &head, Node \* &tail,int data){

Node \*ptr=new Node(data);

if(tail!=NULL){

tail->next=ptr;

ptr->previous=tail;

}

else{

head=ptr;

}

tail=ptr;

}

void TraversalFromHead(Node \* &head){

Node \*temp=head;

while(temp!=NULL){

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

temp=temp->next;

}

cout<<endl;

}

void TraversalFromTail(Node \* &tail){

Node \*temp=tail;

while(temp!=NULL){

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

temp=temp->previous;

}

cout<<endl;

}

void InsertAtIndex(Node \* &head, Node \* &tail,int data,int index){

if(index==0){

InsertionAtHead(head,tail,data);

return;

}

Node \*ptr=new Node(data);

Node \*temp=head;

int i=0;

while(i!=index-1&&temp!=NULL){

temp=temp->next;

i++;}

if(temp->next==NULL){

InsertAtTail(head,tail,data);

return;

}

ptr->next=temp->next;

temp->next=ptr;

ptr->previous=temp;

if(ptr->next!=NULL)

ptr->next->previous=ptr;

}

void ExtremeSwap(Node \*&head,Node \*&tail,int time){

int i=0;

Node \*x=head;

Node \*y=tail;

while(i<time){

if(head==NULL||tail==NULL||head==tail){

return;

}

int temp;

temp=x->data;

x->data=y->data;

y->data=temp;

x=x->next;

y=y->previous;

i++;

cout<<"Values after "<< i <<" Swap: ";

TraversalFromHead(head);

}

}

int main(){

Node \*head=NULL;

Node \*tail=NULL;

InsertionAtHead(head,tail,0);

InsertAtIndex(head,tail,1,1);

InsertAtTail(head,tail,3);

InsertAtIndex(head,tail,2,2);

TraversalFromHead(head);

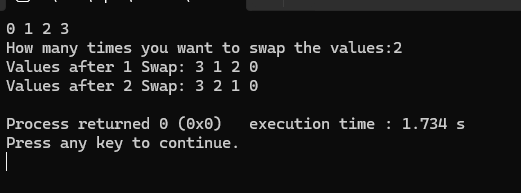
int x;

cout<<"How many times you want to swap the values:";

cin>>x;

ExtremeSwap(head,tail,x);

}



Q7))

Write a program to implement addition of two polynomials. Each node must

contain the value of the coefficient as well as its power as data components.

Sol))

#include<iostream>

using namespace std;

class Node{

public:

int coefficient;

int power;

Node \*next;

Node(int val,int power){

coefficient=val;

next=NULL;

this->power=power;

}

};

void InsertAtHead(Node \*&head,int data,int power){

Node \*ptr=new Node(data,power);

ptr->next=head;

head=ptr;

}

void TraversalFromHead(Node \* &head){

Node \*temp=head;

while(temp!=NULL){

cout<<temp->coefficient<<"x^"<<temp->power;

if(temp->next!=NULL){

cout<<" + ";

}

temp=temp->next;

}

cout<<endl;

}

void InsertAtIndex(Node \* &head,int data,int power,int index){

if(index==0){

InsertAtHead(head,data,power);

return;

}

Node \*ptr=head;

Node \*p=new Node(data,power);

int i=0;

while(i!=index-1 && ptr!=NULL){

ptr=ptr->next;

i++;

}

p->next=ptr->next;

ptr->next=p;

}

void AddPolynomials(Node \*&head1,Node \*&head2){

Node \*temp1=head1;

Node \*temp2=head2;

while(temp1!=NULL){

temp2=head2;

while(temp2!=NULL){

if(temp1->power==temp2->power){

temp1->coefficient+=temp2->coefficient;

}

temp2=temp2->next;

}

temp1=temp1->next;

}

}

int main(){

Node \*head1=NULL;

Node \*head2=NULL;

cout<<"Entering the value for linked list 1: ";

int a,degree;

cout<<"how many degree polynomial equation u want to add in your linked list 1 but (put higher degree than or equal to linked 2 ): ";

cin>>degree;

int i=0;

while(i<=degree){

a=0;

cout<<"enter the coefficient of x^"<<i<<": ";

cin>>a;

InsertAtIndex(head1,a,i,i);

i++;

}

cout<<"Entering the value for Linked list 2: ";

int a1,degree1;

cout<<"how many degree polynomial equation u want to add in your linked list 2: ";

cin>>degree1;

int j=0;

while(j<=degree){

cout<<"Enter the coefficient of x^"<<j<<": ";

cin>>a1;

InsertAtIndex(head2,a1,j,j);

j++;

}

AddPolynomials(head1,head2);

cout<<"LinkedList after adding two polynomials:";

TraversalFromHead(head1);

}

