Task 1

#include<iostream>

using namespace std;

class node{

public:

int data;

int key;

node\* next;

node\* prev;

node(int k, int val){

this->data = val;

this->key = k;

next =NULL;

prev=NULL;

}

};

class LinkedList{

public:

node\* head;

node\* tail;

LinkedList(){

head =NULL;

tail-=NULL;

}

// Inserting at the head

void insertAtHead(int k,int val){

node\* newnode = new node(k,val);

if(head==NULL){

head=newnode;

tail=newnode;

return;

}

newnode->next = head;

head->prev = newnode;

head= newnode;

return;

}

//inserting node at the tail

void insertAtEnd(int k,int val){

node\* newnode =new node(k,val);

if(head==NULL){

head=newnode;

tail=newnode;

return;

}

tail->next = newnode;

newnode->prev=tail;

tail=newnode;

return;

}

//inserting the node after the position

void insertAtPos(int pos,int k,int val){

node\* temp=head;

int count=1;

while(count<(pos-1)){

temp=temp->next;

count++;

}

node\* newnode = new node(k,val);

newnode->next=temp->next;

temp->next = newnode;

newnode->prev = temp;

temp->next->prev=newnode;

}

//Delete the node via key

void deleteViaKey(int k){

node\* temp=head;

while(temp->key!=k){

temp= temp->next;

}

// now we are at the pos of deleting

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

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

}

// Display fun

void display(){

node\* temp=head;

while(temp!=NULL){

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

temp=temp->next;

}

cout<<"Null";

}

};

int main(){

LinkedList newlist;

newlist.insertAtHead(1,3);

newlist.insertAtHead(2,66);

newlist.display();

cout<<endl<<"Inserting(76 ,36 78) at the end of the list"<<endl;

newlist.insertAtEnd(3,76);

newlist.insertAtEnd(4,36);

newlist.insertAtEnd(5,78);

newlist.display();

cout<<endl;

cout<<"Inserting the value(100) at the 3rd position"<<endl;

newlist.insertAtPos(3,6,100);

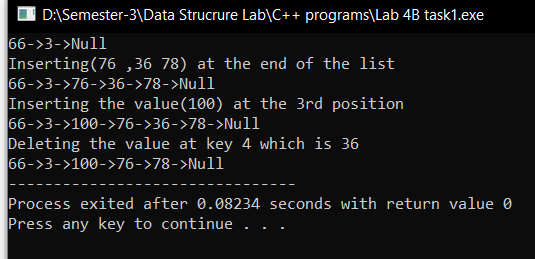
newlist.display();

cout<<endl<<"Deleting the value at key 4 which is 36 "<<endl;

newlist.deleteViaKey(4);

newlist.display();

}



Task 2

#include<iostream>

using namespace std;

class node{

public:

int data;

node\* next;

node\* prev;

node(int val){

this->data = val;

next = NULL;

prev=NULL;

}

};

class cirLinkedList{

public:

node\* head;

node\* tail;

cirLinkedList(){

head = NULL;

tail = NULL;

}

// Adding the node at the end

void insertAtEnd(int val){

node\* newnode = new node(val);

if(head==NULL){

newnode->next= newnode;

newnode->prev = newnode;

head = newnode;

}else{

newnode->next = head;

newnode->prev = head->prev;

newnode->prev->next = newnode;

head->prev=newnode;

}

}

// Inserting the node at the head

void insertAtHead(int val){

node\* newnode = new node(val);

if(head ==NULL){

newnode->next = newnode;

newnode->prev = newnode;

}else{

newnode->next = head;

newnode->prev = head->prev;

head->prev->next = newnode;

head->prev= newnode;

}

head = newnode;

}

// Inserting the node at the Given pos

void insertAtPos(int pos, int val){

node\* temp = head;

int count =1;

while(count<(pos-1)){

temp= temp->next;

count ++;

}

node\* newnode = new node(val);

newnode->next=temp->next;

temp->next= newnode;

newnode->prev=temp;

newnode->prev->next = newnode;

}

//Deleting the node at the end

void deleteAtEnd(){

if(head->next==head){

delete head;

return;

}else{

node\* temp = head->prev;

temp->prev->next = head;

head->prev = temp->prev;

delete temp;

}

}

// display function

void display(){

if(head==NULL){

cout<<"List is Empty"<<endl;

return;

}

node\* temp = head;

do{

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

temp=temp->next;

}while(temp!=head);

cout<<"Null"<<endl;

}

};

int main(){

cirLinkedList list1;

list1.insertAtHead(1);

list1.insertAtHead(2);

list1.insertAtHead(3);

list1.insertAtHead(4);

list1.display();

cout<<endl<<"\nInserting the newnode at the end of the list"<<endl;

list1.insertAtEnd(5);

list1.insertAtEnd(6);

list1.insertAtEnd(7);

list1.insertAtEnd(8);

list1.display();

cout<<endl<<"\nInserting the node at the pos 5 val = 99"<<endl;

list1.insertAtPos(5,99);

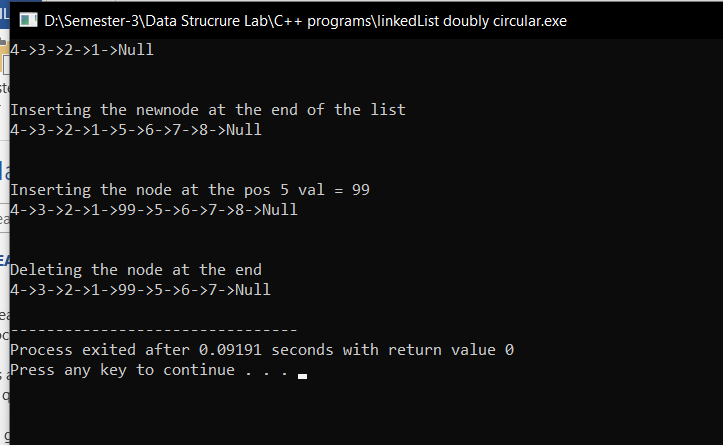
list1.display();

cout<<endl<<"\nDeleting the node at the end"<<endl;

list1.deleteAtEnd();

list1.display();

}



Task 3

#include<iostream>

using namespace std;

class node{

public:

int data;

node\* prev;

node\* next;

node(int val){

this->data = val;

prev = NULL;

next= NULL;

}

};

class doublyLinkedList{

public:

node\* head;

node\* tail;

doublyLinkedList(){

head= NULL;

tail = NULL;

}

// Appending function

void append(int data)

{

node\* newnode = new node(data);

if(head==NULL)

{

head=newnode;

head->next=NULL;

head->prev=NULL;

}

else

{

node \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->prev=temp;

tail=newnode;

}

}

//Inserting at the head

void insertAtHead(int val){

node\* newnode = new node(val);

if(head==NULL){

head= newnode;

tail = newnode;

return;

}

newnode->next =head;

head->prev = newnode;

head = newnode;

return;}

// Display function

void display(){

node\* temp = head;

while(temp!=NULL){

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

temp = temp->next;

}

cout<<"NULL"<<endl;

}

// Concating fun

void concatenate(doublyLinkedList l,doublyLinkedList m)

{

node \*temp=l.head;

while(temp!=NULL)

{

int d=temp->data;

this->append(d);

temp=temp->next;

}

temp=m.head;

while(temp!=NULL)

{

int d=temp->data;

this->append(d);

temp=temp->next;

}

}

};

int main(){

doublyLinkedList L;

L.insertAtHead(1);

L.insertAtHead(2);

L.insertAtHead(3);

L.insertAtHead(4);

L.display();

doublyLinkedList M;

M.insertAtHead(5);

M.insertAtHead(6);

M.insertAtHead(7);

M.insertAtHead(8);

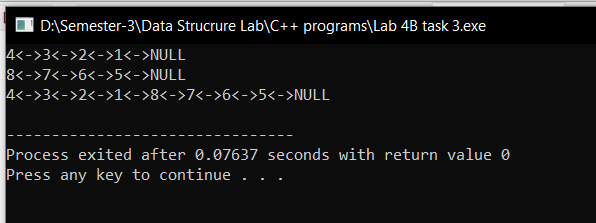
M.display();

doublyLinkedList newlist;

newlist.concatenate(L,M);

newlist.display();

}



Task 3

#include<iostream>

using namespace std;

class node{

public:

int data;

node\* next;

node\* prev;

node()

{

next=NULL;

}

node(int a)

{

data=a;

next=NULL;

prev=NULL;

}

};

class linkedlist{

public:

node\* head;

linkedlist()

{

head=NULL;

}

void append(int data)

{

node\* newnode = new node(data);

if(head==NULL)

{

head=newnode;

head->next=head;

}

else

{

node \*temp=head;

do

{

temp=temp->next;

}while(temp->next!=head);

temp->next=newnode;

newnode->prev=temp;

newnode->next=head;

head->prev=newnode;

}

}

void swap(int a,int b)

{

node \*temp=head;

int count=0;

while(count!=a-1)

{

count++;

temp=temp->next;

}

node \*n1=temp;

temp=head;

count=0;

while(count!=b-1)

{

count++;

temp=temp->next;

}

node \*n2=temp;

temp=n1->next;

n1->next=n2->next;

n2->next=temp;

temp=n1->prev;

n1->prev=n2->prev;

n2->prev=temp;

n1->next->prev=n1;

n2->next->prev=n2;

n1->prev->next=n1;

n2->prev->next=n2;

}

void display()

{

node \*temp=head;

do

{

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

temp=temp->next;

}while(temp!=head);

}

};

main()

{

linkedlist list;

cout<<"Appending to list:\n";

list.append(1);

list.append(7);

list.append(4);

list.append(2);

list.append(6);

list.append(4);

list.append(5);

list.append(3);

list.append(9);

list.append(8);

list.display();

int a,b;

cout<<"\nEnter the Positions of list: ",cin>>a>>b;

list.swap(a,b);

list.display();

}

