Lab Questions:

Q1. You are given an empty singly linked list. Assume that this list can contain

whole numbers only. Write functions to

a. Insert ‘n’ number of data in the singly linked list. Insert from the head.

Practice this question using Virtual Lab

(https://ds1-iiith.vlabs.ac.in/exp/linked-list/singly-linked-

list/sllexercise.html)

b. Find the total number of nodes in the linked list, and give their average.

c. Print first ‘m’ data from the linked list. Assume that ‘m’ is less than ‘n’.

Example:

Input:

Linked list: {1, 3, -9, 45, 2, 3, 56, 100, -67}

m=4

Output: {1, 3, -9, 45, 2}

Example:

Input:

Linked list: {1, 3, -9, 45, 2, 3, 56, 100, -67}

m=10

Output: Incorrect value of m

d. Find the middle element of the linked list and check if it’s odd or even. Print

an appropriate output.

Example:

Input:

Linked list: {1, 3, -9, 45, 2, 3, 56, 100, -67}

Output: 3 is odd

e. Find the ‘l’ number from the end of the list.

Example:

Input:

Linked list: {1, 3, -9, 45, 2, 3, 56, 100, -67}

l=3

Output: {56, 100, -67}

f. Find if a given number exists in the list. If it does, write function to delete it.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100, -67}

Number to be found: 45

Output: 45 exists in the original list

Final list: {1, 3, -9, 2, 3, 56, 100, -67}

If the value exists multiple times, delete only the first instance.

g. Interchange a pair of values with another given pair in the linked list.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100, -67}

Pairs to be exchanged: {1,3} with {56,100}

Output: {56, 100, -9, 45, 2, 3, 1, 3, -67}

Hint: first check if the pair exists and then apply interchange function. If

multiple or duplicate pairs are found, consider the first instance of the pair.

h. Check whether a given sub-list exists in the given linked list. If it exists, give

its position (i.e., the staring position of the sub-list in the master linked list).

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100, -67}

Sub-list to be: {3, -9, 45}

Output: Exists at position 2.

{1, 3,-9, 45, 2, 3, 56, 100, -67}

Assumption: consider only the first occurrence of the sub-list.

i. Reverse a sub-list in the given linked list.

Example:

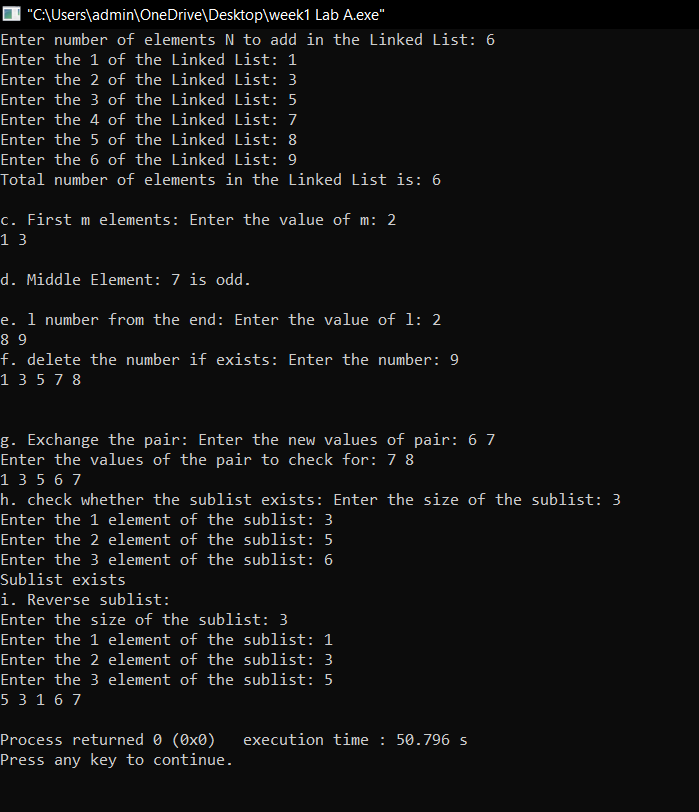
Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100, -67}

Sub-list to be reversed: {3, -9, 45}

Output: {1, 45,-9, 3, 2, 3, 56, 100, -67}

Assume that the user inputs the sub-list is found in the master linked list.



#include <iostream>

#include <vector>

#include <bits/stdc++.h>

using namespace std;

class node

{

public:

int data;

node \*next;

node()

{

data=0;

next=nullptr;

}

node(int n)

{

data=n;

next=nullptr;

}

};

/\*

node\* convertarr(vector<int> &arr)

{

node\* head=new node(arr[0]);

node\* mover=head;

for(int i=1;i<arr.size();i++)

{

node\* temp=new node(arr[i]);

mover->next=temp;

mover=temp;

}

return head;

}

\*/

node\* insertn(int n)

{

vector<int> arr;

int ele;

for(int i=0;i<n;i++)

{

cout<<"Enter the "<<i+1<<" of the Linked List: ";

cin>>ele;

arr.push\_back(ele);

}

node\* head=new node(arr[0]);

node\* mover=head;

for(int i=1;i<arr.size();i++)

{

node\* temp=new node(arr[i]);

mover->next=temp;

mover=temp;

}

return head;

}

int noofelements(node \* head)

{

node\* temp=head;

int count=0;

while(temp!=nullptr)

{

temp=temp->next;

count++;

}

return count;

}

void mdata(node \*head)

{

int m;

int n=noofelements(head);

cout<<"Enter the value of m: ";

cin>>m;

if(m>n)

{

cout<<"Invalid Input!!"<<endl;

return;

}

int count=0;

node\* temp=head;

while(count!=m)

{

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

temp=temp->next;

count++;

}

cout<<endl<<endl;

}

void midelement(node \*head)

{

int n=noofelements(head);

int mid=n/2;

int count=0;

node\* temp=head;

while(count!=mid)

{

temp=temp->next;

count++;

}

int val=temp->data;

if(val%2==0)

{

cout<<val<<" is even."<<endl;

}

else cout<<val<<" is odd."<<endl;

cout<<endl;

}

void lelements(node \* head)

{

int l;

cout<<"Enter the value of l: ";

cin>>l;

int n=noofelements(head);

int m=n-l;

int count=0;

node\* temp=head;

while(count!=m)

{

temp=temp->next;

count++;

}

while(temp!=nullptr)

{

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

temp=temp->next;

}

cout<<endl;

}

node\* deletenode(node \* head)

{

int num;

cout<<"Enter the number: ";

cin>>num;

if((head->data)==num)

{

node \* temp=head;

head=head->next;

delete temp;

return head;

}

node\* temp=head;

while(temp!=NULL)

{

node\* prev=temp;

temp=temp->next;

if((temp->data)==num)

{

prev->next=temp->next;

delete temp;

return head;

}

}

return head;

}

node \* pairchange(node \* head,pair<int,int> d)

{

int a,b;

cout<<"Enter the values of the pair to check for: ";

cin>>a>>b;

node\* temp=head;

while(temp!=NULL)

{

if((temp->data==a)&&(temp->next->data)==b)

{

temp->data=d.first;

temp->next->data=d.second;

return head;

}

temp=temp->next;

}

return head;

}

void display(node\* head)

{

node\* temp=head;

while(temp!=NULL)

{

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

temp=temp->next;

}

cout<<endl;

}

bool sublist(node\* head)

{

int n=noofelements(head);

int m;

cout<<"Enter the size of the sublist: ";

cin>>m;

if(m>n)

{

cout<<"Invalid Size Input: ";

sublist(head);

}

int \*arr=new int[m];

for(int i=0;i<m;i++)

{

cout<<"Enter the "<<i+1<<" element of the sublist: ";

cin>>arr[i];

}

node\* temp=head;

int i=0;

while(temp!=NULL)

{

if((temp->data)==arr[0])

{

while((temp->next!=NULL)&&(i!=m-1))

{

if((temp->data)!=arr[i]) break;

temp=temp->next;i++;

}

if(i==m-1) return true;

i=0;

}

temp=temp->next;

}

return false;

}

void reverselist(node\* head)

{

int n=noofelements(head);

int m,ele;

cout<<"Enter the size of the sublist: ";

cin>>m;

vector<int> arr;

for(int i=0;i<m;i++)

{

cout<<"Enter the "<<i+1<<" element of the sublist: ";

cin>>ele;

arr.push\_back(ele);

}

node\* temp=head;

int i=0;

while(temp!=NULL)

{

if((temp->data)==arr[0])

{

node\* mover=temp;

while((temp->next!=NULL)&&(i!=m-1))

{

if((temp->data)!=arr[i]) break;

temp=temp->next;i++;

}

if(i==m-1)

{

i=0;

reverse(arr.begin(),arr.end());

while(i!=m)

{

mover->data=arr[i];

mover=mover->next;

i++;

}

}

}

temp=temp->next;

}

}

int main()

{

int n;

cout<<"Enter number of elements N to add in the Linked List: ";

cin>>n;

node\* listt=insertn(n);

cout<<"Total number of elements in the Linked List is: "<<noofelements(listt)<<endl<<endl;

cout<<"c. First m elements: ";

mdata(listt);

cout<<"d. Middle Element: ";

midelement(listt);

cout<<"e. l number from the end: ";

lelements(listt);

cout<<"f. delete the number if exists: ";

deletenode(listt);

display(listt);

cout<<endl<<endl<<"g. Exchange the pair: ";

pair<int,int> d;

cout<<"Enter the new values of pair: ";

cin>>d.first;

cin>>d.second;

pairchange(listt,d);

display(listt);

cout<<"h. check whether the sublist exists: ";

if(sublist(listt))

{

cout<<"Sublist exists"<<endl;

}

else

{

cout<<"Sublist does not exist"<<endl;

}

cout<<"i. Reverse sublist: "<<endl;

reverselist(listt);

display(listt);

}

Q2. Assume that you have a linked list that can contain strings, i.e., each node can

contain a string. Write a function to:

a. Print all the nodes in the linked list

b. Print all the strings (node values) that start with a particular alphabet.

c. Find if a given string exists in the linked list or not. Give appropriate output

message.

d. Find the string with maximum length.

e. Find if a node contains “xyz” as a sub-string or not. Give appropriate output

message.

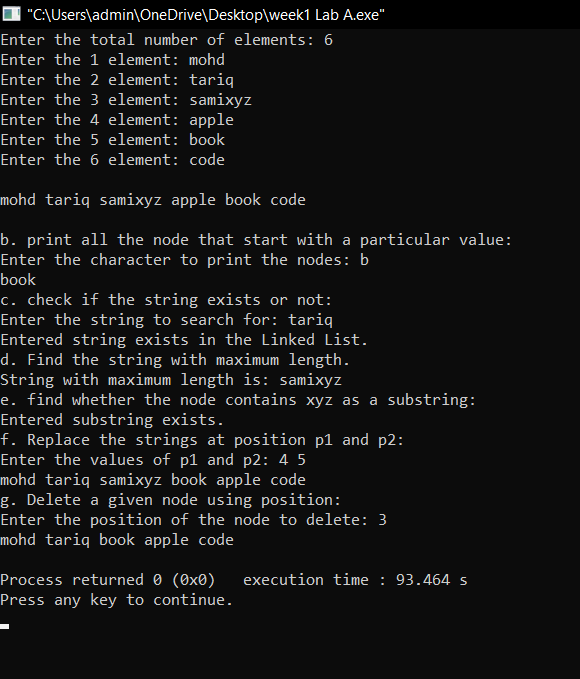
f. Interchange the strings given in the positions p1, p2. These positions are

user input. Check conditions that both p1 and p2 position exist in the given

linked list, eg: suppose that your linked list consists of 4 strings only, and if

user given p1=7, p2 = 10, then error message must be generated.

g. Delete a given node (either by value or by position).



class node

{

public:

string data;

node \*next;

node()

{

data="";

next=nullptr;

}

node(string n)

{

data=n;

next=nullptr;

}

};

node\* create()

{

int n;

cout<<"Enter the total number of elements: ";

cin>>n;

string\* arr=new string[n];

for(int i=0;i<n;i++)

{

cout<<"Enter the "<<i+1<<" element: ";

cin>>arr[i];

cin.ignore();

cin.clear();

}

node\* head=new node(arr[0]);

node\* mover=head;

for(int i=1;i<n;i++)

{

node\* temp=new node(arr[i]);

mover->next=temp;

mover=temp;

}

return head;

}

int noofelements(node \* head)

{

node\* temp=head;

int count=0;

while(temp!=nullptr)

{

temp=temp->next;

count++;

}

return count;

}

void display(node\* head)

{

node\* temp=head;

while(temp!=NULL)

{

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

temp=temp->next;

}

cout<<endl;

}

void printp(node\* head)

{

char a;

cout<<"Enter the character to print the nodes: ";

cin>>a;

node\* temp=head;

while(temp!=NULL)

{

if((temp->data)[0]==a)

{

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

}

temp=temp->next;

}

cout<<endl;

}

bool exists(node \* head)

{

string str;

cout<<"Enter the string to search for: ";

cin>>str;

node\* temp=head;

while(temp!=NULL)

{

if(temp->data==str)

{

return true;

}

temp=temp->next;

}

return false;

}

void maxstring(node\* head)

{

int max=0;

string str;

node\* temp=head;

while(temp!=NULL)

{

if(((temp->data).length())>max)

{

max=(temp->data).length();

str=temp->data;

}

temp=temp->next;

}

cout<<"String with maximum length is: "<<str<<endl;

}

bool substring(node\* head)

{

string str="xyz";

node\* temp=head;

while(temp!=NULL)

{

if((temp->data).find(str)!=std::string::npos)

{

return (temp->data).find(str);

}

temp=temp->next;

}

return false;

}

node\* deletenode(node\* head)

{

if(head==NULL)

{

return NULL;

}

int k;

cout<<"Enter the position of the node to delete: ";

cin>>k;

if(k>noofelements(head))

{

cout<<"Invalid input: "<<endl;

deletenode(head);

}

if(k==1)

{

node\* temp=head;

head=head->next;

delete temp;

return head;

}

int count=1;

node\* temp=head->next;

node\* prev=head;

while(temp!=NULL)

{

count++;

if(count==k)

{

prev->next=temp->next;

delete temp;

return head;

}

prev=temp;

temp=temp->next;

}

return head;

}

void replacestr(node\* head)

{

int p1,p2;

int n=noofelements(head);

cout<<"Enter the values of p1 and p2: ";

cin>>p1>>p2;

if(p1>n | p2>n)

{

cout<<"invalid Input: p1 and p2 can't be greater than the length of linked list"<<endl;

replacestr(head);

}

int count=0;

node\* temp=head;

string str1,str2;

while(temp!=NULL)

{

count++;

if(count==p1)

{

str1=temp->data;

}

if(count==p2)

{

str2=temp->data;

}

temp=temp->next;

}

node\* temp2=head;

count=0;

while(temp2!=NULL)

{

count++;

if(count==p1)

{

temp2->data=str2;

}

if(count==p2)

{

temp2->data=str1;

}

temp2=temp2->next;

}

}

int main()

{

node\* head=create();

cout<<endl;

display(head);

cout<<endl;

cout<<"b. print all the node that start with a particular value: "<<endl;

printp(head);

cout<<"c. check if the string exists or not: "<<endl;

if(exists(head))

{

cout<<"Entered string exists in the Linked List. "<<endl;

}

else cout<<"Entered string does not exist. "<<endl;

cout<<"d. Find the string with maximum length. "<<endl;

maxstring(head);

cout<<"e. find whether the node contains xyz as a substring: "<<endl;

if(substring(head))

{

cout<<"Entered substring exists. "<<endl;

}

else

{

cout<<"Entered substring does not exist. "<<endl;

}

cout<<"f. Replace the strings at position p1 and p2: "<<endl;

replacestr(head);

display(head);

cout<<"g. Delete a given node using position: "<<endl;

node\* newlist=deletenode(head);

display(newlist);

}

Q3. Implement a circular linked list that can contain integer elements. Add

functions to:

a. Insert elements.

b. Print elements

c. Count the number of elements

d. Find if any element has a negative value.

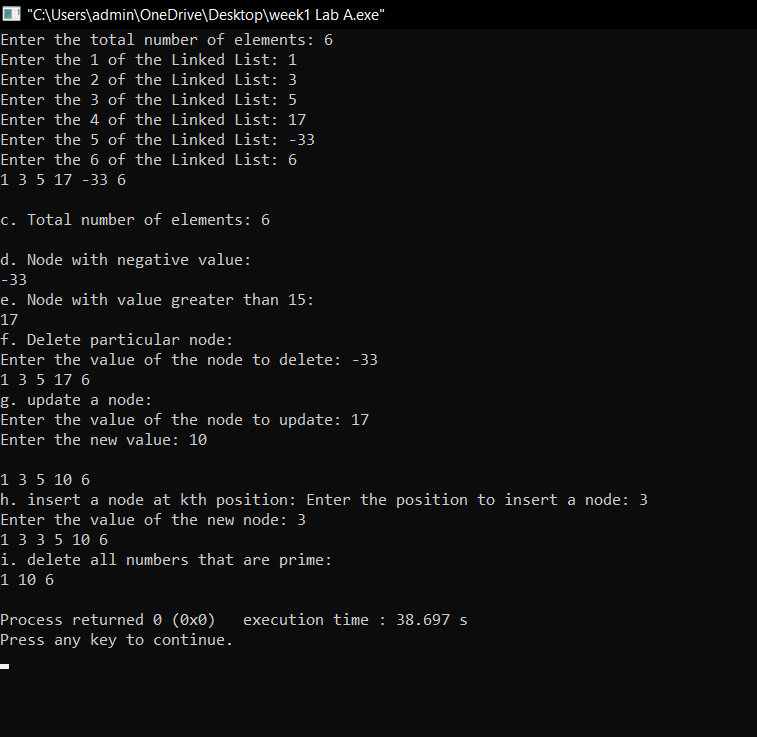
e. Find the number of nodes having a value greater than 15.

f. Delete a particular element from the list.

g. Update the value of a particular element.

h. Insert a value at a given position.

i. Delete all nodes that have a prime number as their value.



class node

{

public:

int data;

node\* next;

node(int n)

{

data=n;

next=nullptr;

}

};

node\* create()

{

int n;

cout<<"Enter the total number of elements: ";

cin>>n;

vector<int> arr;

int ele;

for(int i=0;i<n;i++)

{

cout<<"Enter the "<<i+1<<" of the Linked List: ";

cin>>ele;

arr.push\_back(ele);

}

node\* head=new node(arr[0]);

node\* mover=head;

for(int i=1;i<arr.size();i++)

{

node\* temp=new node(arr[i]);

mover->next=temp;

mover=temp;

}

mover->next=head;

return head;

}

void display(node\* head)

{

node\* temp=head;

node\* stop=head;

do

{

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

temp=temp->next;

}while(temp!=stop);

cout<<endl;

}

int noofelements(node\* head)

{

int count=0;

node\* temp=head;

node\* stop=head;

do

{

temp=temp->next;

count++;

}while(temp!=stop);

return count;

}

void insertelement(node\* head)

{

int k,val;

cout<<"Enter the position to insert a node: ";

cin>>k;

cout<<"Enter the value of the new node: ";

cin>>val;

if(k==1)

{

node\* temp=new node(val);

node\* tail=head;

while(tail->next!=head)

{

tail=tail->next;

}

temp->next=head;

head=temp;

tail->next=head;

}

int count=1;

node\* temp=head;

do

{

node\* prev=temp;

temp=temp->next;

count++;

if(count==k)

{

node\* ele=new node(val);

prev->next=ele;

ele->next=temp;

}

}while(temp!=head);

}

void negativenodes(node\* head)

{

node\* temp=head;

node\* stop=head;

do

{

if(temp->data<0)

{

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

}

temp=temp->next;

}while(temp!=stop);

cout<<endl;

}

void greaterthan15(node\* head)

{

node\* temp=head;

node\* stop=head;

do

{

if(temp->data>15)

{

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

}

temp=temp->next;

}while(temp!=stop);

cout<<endl;

}

node\* deletenode(node\* head)

{

int val;

cout<<"Enter the value of the node to delete: ";

cin>>val;

node\* temp=head;

node\* stop=head;

if(temp->data==val)

{

node\* mover=temp;

head=head->next;

delete mover;

return head;

}

do

{

node\* prev=temp;

temp=temp->next;

if(temp->data==val)

{

prev->next=temp->next;

delete temp;

return head;

}

}while(temp!=stop);

cout<<endl;

}

node\* updatenode(node\* head)

{

int val,newval;

cout<<"Enter the value of the node to update: ";

cin>>val;

cout<<"Enter the new value: ";

cin>>newval;

node\* temp=head;

node\* stop=head;

do

{

if(temp->data==val)

{

temp->data=newval;

}

temp=temp->next;

}while(temp!=stop);

cout<<endl;

}

bool isprime(int n)

{

if(n<=1) return false;

int m=pow(n,0.5);

for(int i=2;i<=m;i++)

{

if(n%i==0) return false;

}

return true;

}

void deleteprime(node\* head)

{

node\* temp=head;

node\* prev=nullptr;

do

{

if(isprime(temp->data))

{

if(prev)

{

prev->next=temp->next;

}

else

{

node\* tail=head;

while(tail!=head) tail=tail->next;

head=temp->next;

tail->next=head;

}

node\* todelete=temp;

temp=temp->next;

delete todelete;

if(temp==head) break;

}

else

{

prev=temp;

temp=temp->next;

}

}while(temp!=head);

cout<<endl;

}

int main()

{

node\* head=create();

display(head);

cout<<endl<<"c. Total number of elements: ";

cout<<noofelements(head)<<endl<<endl;

cout<<"d. Node with negative value: "<<endl;

negativenodes(head);

cout<<"e. Node with value greater than 15: "<<endl;

greaterthan15(head);

cout<<"f. Delete particular node: "<<endl;

node\* deleted=deletenode(head);

display(deleted);

cout<<"g. update a node: "<<endl;

updatenode(head);

display(head);

cout<<"h. insert a node at kth position: ";

insertelement(head);

display(head);

cout<<"i. delete all numbers that are prime: ";

deleteprime(head);

display(head);

}

Q4. Create an empty doubly linked list to store integers. Perform the following by

writing appropriate functions to:

a. Insert and print elements of the list.

b. Traverse all nodes and check if the value is divisible by a number ‘m’.

c. Delete all the nodes from the list that are greater than the given value ‘x’.

d. Find the number of elements between two duplicate values.

Example:

Input:

Doubly Linked list: {1, 3, -9, 45, 2, -56, 3, 56, 100, -67, 3, 3}

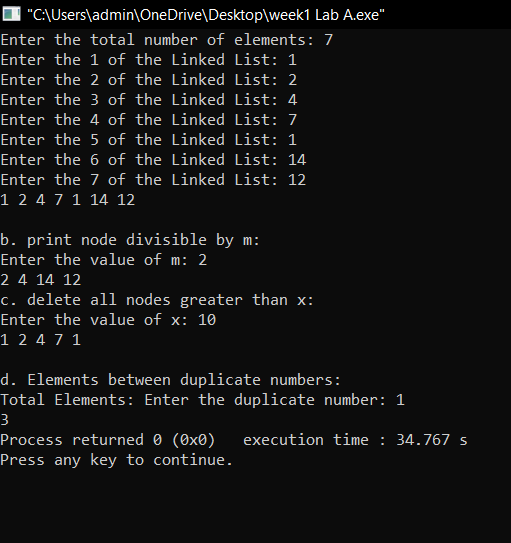
Duplicate element: 3

Output: No. of elements between a pair of ‘3’ = 4.

Assumption: You are considering only the first instance of duplicity, i.e.,

between {3, -9, 45, 2, -56, 3} and not for instances like {3, 56, 100, -67, 3} nor

for {3, 3} or any others.



class node

{

public:

int data;

node\* next;

node\* prev;

node(int n)

{

data=n;

prev=nullptr;

next=nullptr;

}

};

node\* create()

{

int n;

cout<<"Enter the total number of elements: ";

cin>>n;

vector<int> arr;

int ele;

for(int i=0;i<n;i++)

{

cout<<"Enter the "<<i+1<<" of the Linked List: ";

cin>>ele;

arr.push\_back(ele);

}

node\* head=new node(arr[0]);

node\* mover=head;

for(int i=1;i<arr.size();i++)

{

node\* temp=new node(arr[i]);

temp->prev=mover;

mover->next=temp;

mover=temp;

}

return head;

}

void display(node\* head)

{

node\* temp=head;

while(temp!=NULL)

{

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

temp=temp->next;

}

cout<<endl;

}

void divisiblebym(node\* head)

{

int m;

cout<<"Enter the value of m: ";

cin>>m;

if(m==0)

{

cout<<"Invalid Input"<<endl;

divisiblebym(head);

}

node\* temp=head;

while(temp!=NULL)

{

if((temp->data)%m==0)

{

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

}

temp=temp->next;

}

cout<<endl;

}

int noofelements(node\* head)

{

node\* temp=head;

int count=0;

while(temp!=NULL)

{

count++;

temp=temp->next;

}

return count;

}

void deletenode(node\* head)

{

int x;

cout<<"Enter the value of x: ";

cin>>x;

node\* temp=head;

while(temp!=nullptr)

{

if(temp->data>x)

{

node\* toDelete=temp;

if(temp->prev)

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

else

head=temp->next;

if(temp->next)

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

else

node\* tail=temp->prev;

temp=temp->next;

delete toDelete;

}

else

temp=temp->next;

}

}

int duplicates(node\* head)

{

int val;

cout<<"Enter the duplicate number: ";

cin>>val;

node\* first=nullptr;

node\* second=nullptr;

node\* temp=head;

while(temp!=nullptr)

{

if(temp->data==val)

{

if(first==nullptr)

first=temp;

else

{

second=temp;

break;

}

}

temp=temp->next;

}

if(first==nullptr||second==nullptr)

return -1;

int count=0;

temp=first->next;

while(temp!=second)

{

count++;

temp=temp->next;

}

return count;

}

int main()

{

node\* head=create();

display(head);

cout<<endl;

cout<<"b. print node divisible by m: "<<endl;

divisiblebym(head);

cout<<"c. delete all nodes greater than x: "<<endl;

deletenode(head);

display(head);

cout<<endl;

cout<<"d. Elements between duplicate numbers: "<<endl;

cout<<"Total Elements: "<<duplicates(head);

}