24F-0040

Laiba

Lab-04

Week-04

**Task-01**

#include<iostream>

using namespace std;

//---------------------------------------------------------------------------

class Node{

public:

int data;

Node\* next;

Node\* prev;

Node(int val){

data=val;

prev=next=nullptr;

}

};

Node\* head=nullptr;

//----------------------------------------------------------------------------

void Traversal(Node\* head){

cout<<"\n Doubly linked list: ";

Node\* temp=head;

while (temp!=nullptr){

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

temp=temp->next;

}cout<<endl;

}

//-----------------------------------------------------------------------------

void insertion(Node\* head){

int index=0,value=0,chk=1;

Node\* new\_node=new Node(value);

cout<<"============================================";

cout<<"\n Enter a value to insert at beginning: ";

cin>>value;

//insertion at beginning-------------------------------

if(head==nullptr){

head=new\_node;

}else{

new\_node->next=head;

head->prev=new\_node;

head=new\_node;

}

cout<<"\n After insertion at beginning: ";

Traversal(head);

cout<<"==============================================";

//insertion in middle----------------------------------

cout<<"\n Enter a value to insert in middle: ";

cin>>value;

while(chk!=0){

cout<<"\n Enter at which index you wanna insert value: ";

cin>>index;

if(index<0 || index>5){

chk=1;

}else{

chk=0;

}

}

Node\* new\_node2=new Node(value);

Node\* temp2=head;

Node\* node2=head->next;

for(int i=0;i<5;i++){

if((i+1)==index){

new\_node2->next=node2;

new\_node2->prev=node2->prev;

node2->prev->next=new\_node2;

node2->prev=new\_node2;

}

temp2=temp2->next;

node2=node2->next;

}

cout<<"\n After adding a new node in miidle: ";

Traversal(head);

cout<<"====================================================";

//insertion at end;

value=0;

Node\* temp3=head;

cout<<"\n Enter value to add in the end: ";

cin>>value;

Node\* new\_node3=new Node(value);

while(temp3!=nullptr){

if(temp3->next==nullptr){

new\_node3->prev=temp3;

temp3->next=new\_node3;

new\_node3->next=nullptr;

}

temp3=temp3->next;

}

cout<<"\n After adding node at end: ";

Traversal(head);

cout<<"======================================================";

}

//----------------------------------------------------------------------------

void deletion(Node\* head){

cout<<"====================================================";

//deletion from beginning

if(head==nullptr){

cout<<"\n List is already empty!"<<endl;

}

else{

Node\* tempd1=head;

head=head->next;

delete tempd1;

cout<<"\n List after deleting 1st element is: "<<endl;

Traversal(head);

}

cout<<"=====================================================";

//deletion in middle

int index,lth;

cout<<"\n Enter which index element you wanna delete: ";

cin>>index;

Node\* tempd2=head;

for (int i=0;i<5;i++){

if(i==index){

tempd2->next->prev=tempd2->prev;

tempd2->prev->next=tempd2->next;

delete tempd2;

break;

}

tempd2=tempd2->next;

}

cout<<"\n After deleting an element from middle: ";

Traversal(head);

cout<<"======================================================";

//delete from end

Node\* tempd3=head;

Node\* extra;

while(tempd3->next!=nullptr){

tempd3=tempd3->next;

}

extra=tempd3->prev;

extra->next=nullptr;

delete tempd3;

cout<<"\n After deleting last element of list: ";

Traversal(head);

cout<<"=======================================================";

}

//------------------------------------------------------------------------------

void looping(Node\* head){

Node\* templ1=head;

int size=0;

cout<<"\n Enter number of elements you wanna see in looping: ";

cin>>size;

for(int i=0;i<size;i++){

cout<<templ1->data<<" ";

templ1=templ1->next;

if(templ1->next==nullptr){

templ1->next=head;

head->prev=templ1;

}

}

cout<<endl;

cout<<"\n==========================================================";

}

//----------------------------------------------------------------------------

int main(){

Node\* head=new Node(1);

Node\* second=new Node(2);

Node\* third=new Node(4);

Node\* forth=new Node(5);

Node\* fifth=new Node(6);

head->next=second;

second->next=third;

third->next=forth;

forth->next=fifth;

fifth->next=nullptr;

fifth->prev=forth;

forth->prev=third;

third->prev=second;

second->prev=head;

Traversal(head);

int choice;

cout<<"====================================================";

while(choice!=0){

cout<<"Enter your choice: "<<endl;

cout<<"1)Insertion \n2)Deletion \n3)Looping of list\n0)End programme"<<endl;

cout<<"\n Choice: ";

cin>>choice;

switch (choice){

case(1):{

insertion(head);

break;

}

case(2):{

deletion(head);

break;

}

case(3):{

looping(head);

break;

}

default:{

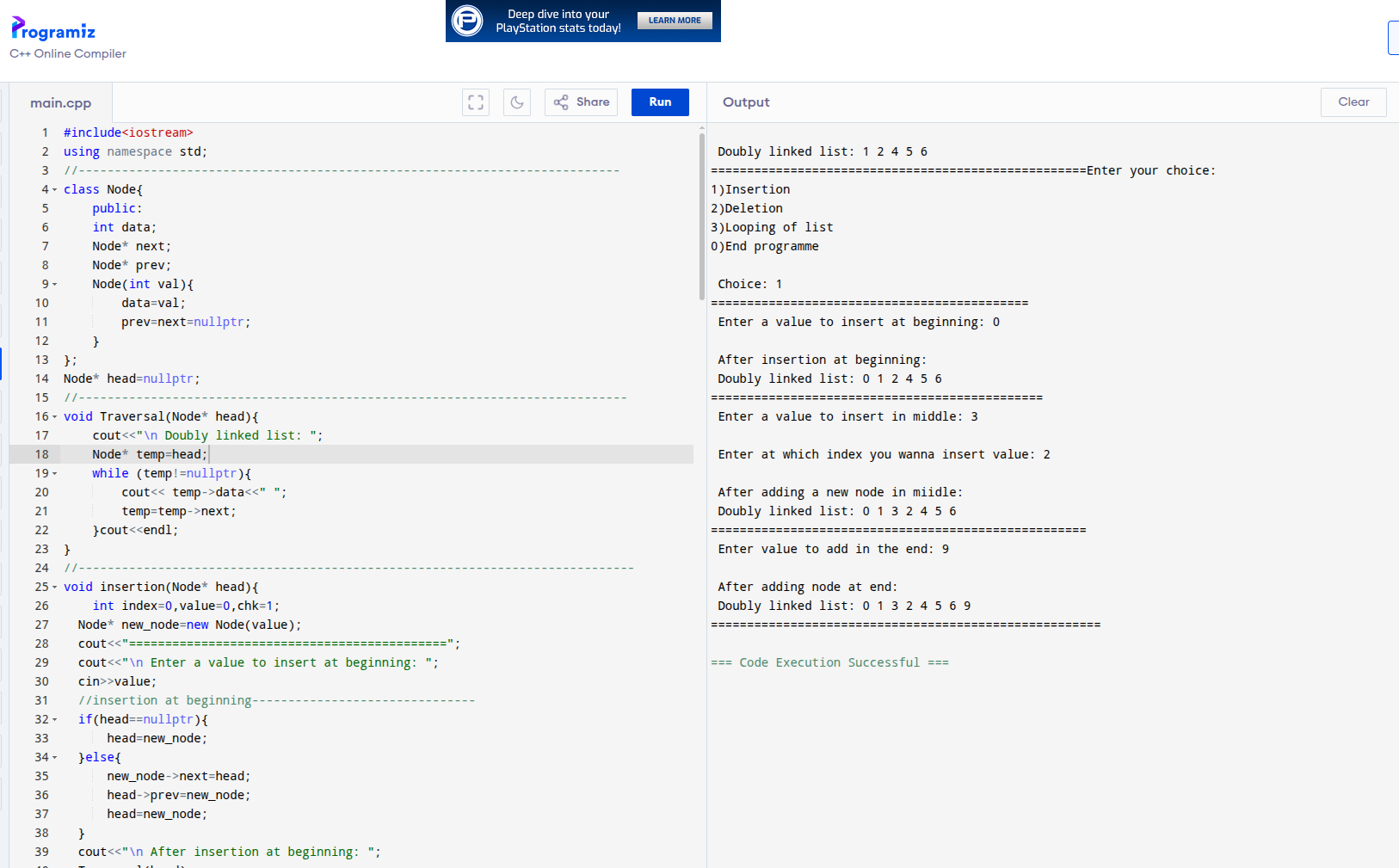
cout<<"Invalid choice!"<<endl;

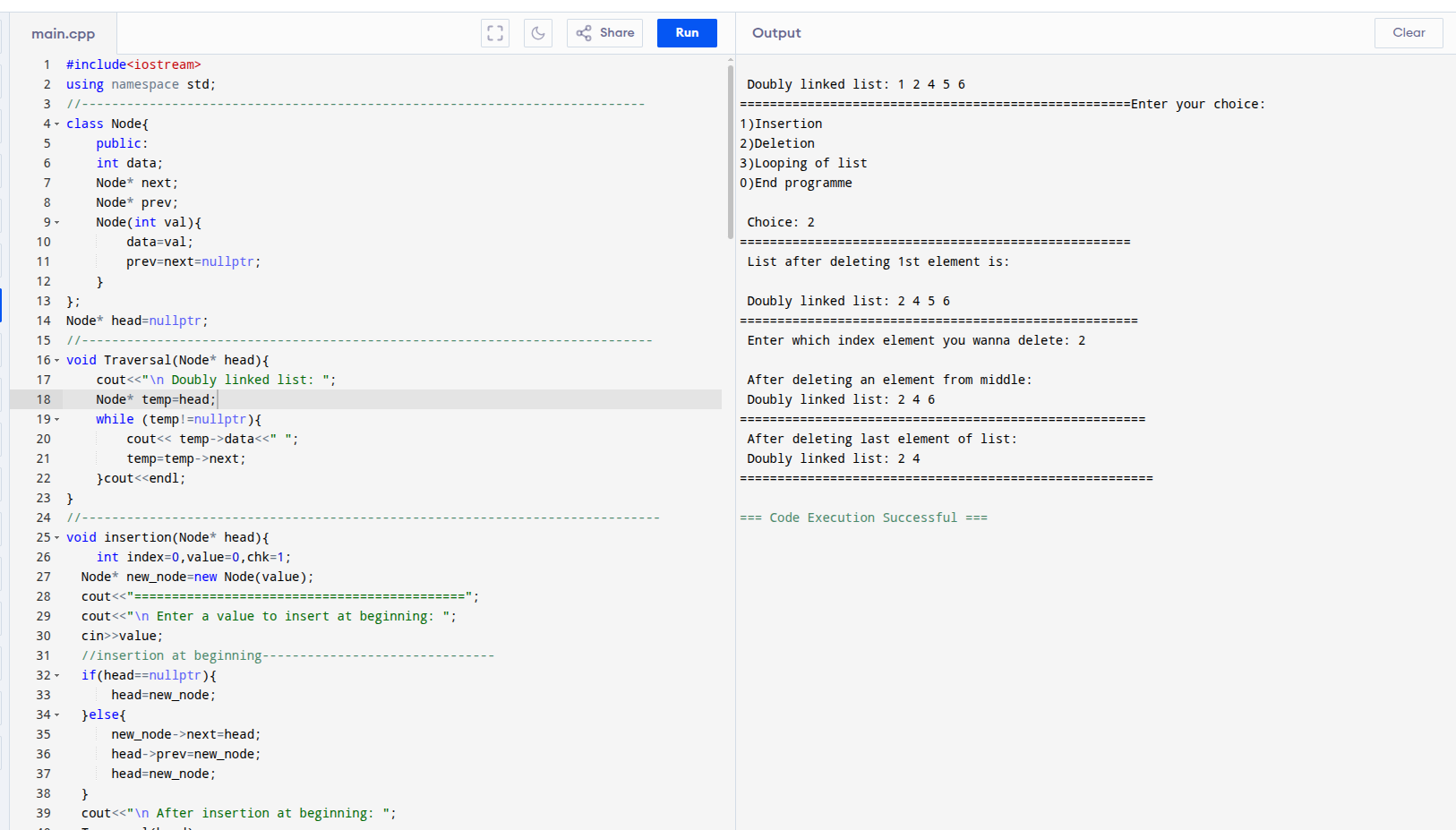
}

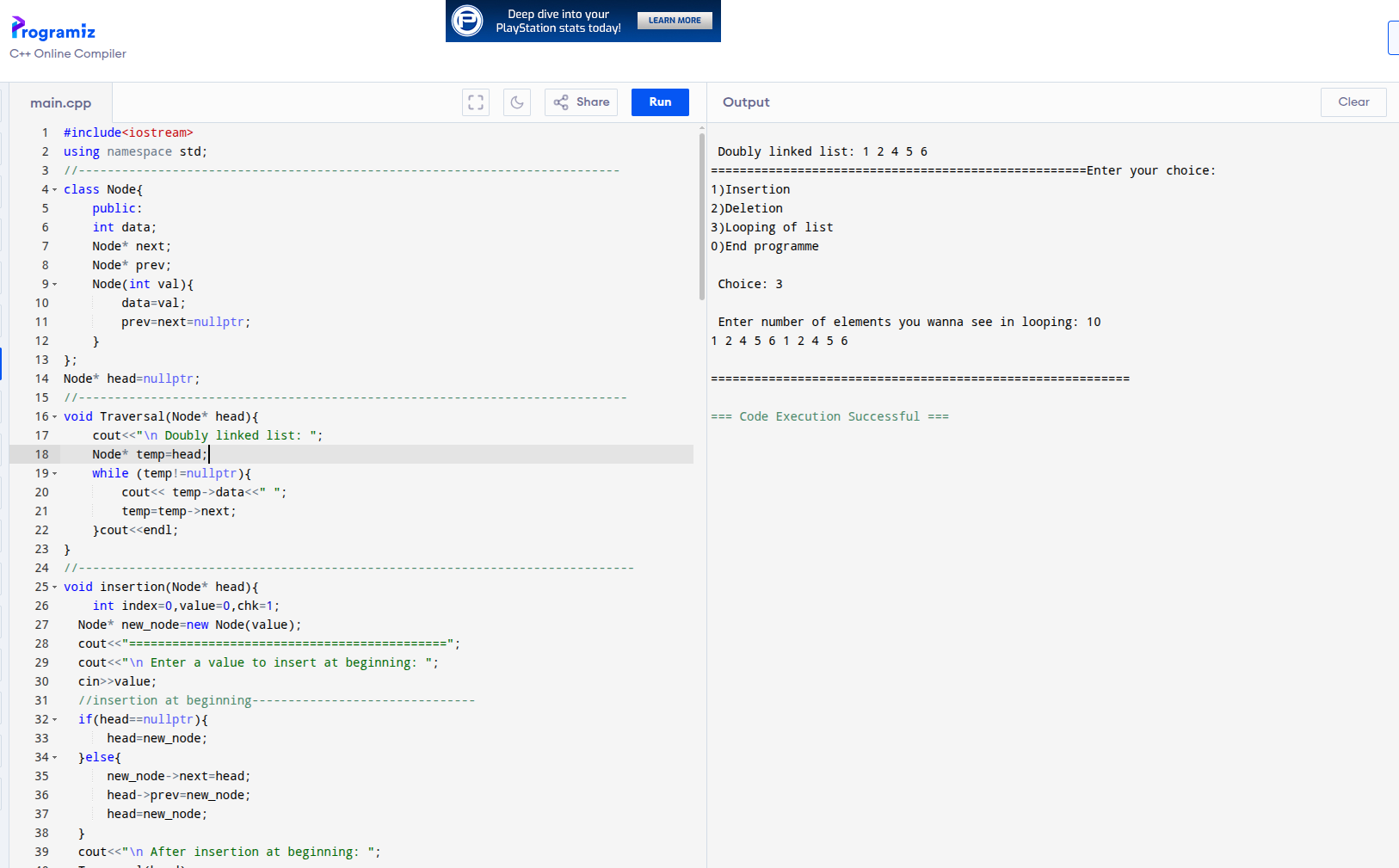
}

return 0;

}}







**Task-02**

#include<iostream>

using namespace std;

//-------------------------------------------------------------------------

template <typename T>

class Node{

public:

T data;

Node<T>\* next;

Node(T value){

this->data=value;

this->next=nullptr;

}

};

//----------------------------------------------------------------------

template<typename T>

void Traversal(Node<T>\* head){

cout<<"\n================================================="<<endl;

Node<T>\* temp=head;

do{

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

temp=temp->next;

}while(temp!=head);

cout<<endl;

}

//---------------------------------------------------------------------------

template <typename T>

void insertion(Node<T>\* head){

cout<<"\n================================================="<<endl;

//insertion at beginning

int value=0;

cout<<"Enter value to insert in the beginning of list: ";

cin>>value;

Node<T>\* new\_node=new Node<T>(value);

if(head==nullptr){

head=new\_node;

new\_node->next=head;

}else{

Node<T>\* temp=head;

while(temp->next!=head){

temp=temp->next;

}

new\_node->next=head;

temp->next=new\_node;

head=new\_node;

cout<<"\n After inserting value at beginning: ";

Traversal(head);

cout<<"\n============================================"<<endl;

}

cout<<"\n================================================"<<endl;

//insert in middle

int value2=0,index=0,i=0;

cout<<"Enter a value to add in the middle: ";

cin>>value2;

cout<<"Enter at which index you wanna add the value: ";

cin>>index;

Node<T>\* new\_node2=new Node<T>(value2);

Node<T>\* temp2=head;

while(temp2->next!=head){

if(i==index){

new\_node2->next=temp2->next;

temp2->next=new\_node2;

}i++;

temp2=temp2->next;

}

cout<<"\n After inserting element in middle of list: ";

Traversal(head);

cout<<"\n============================================="<<endl;

cout<<"\n================================================="<<endl;

//insert at end

int value3=0;

cout<<"Enter a value to add in the end: ";

cin>>value3;

Node<T>\* new\_node3=new Node<T>(value3);

Node<T>\* temp3=head;

while(temp3->next!=head){

temp3=temp3->next;

}

temp3->next=new\_node3;

new\_node3->next=head;

cout<<"\n After inserting at end of list: ";

Traversal(head);

cout<<"\n======================================================"<<endl;

}

//---------------------------------------------------------------------

template <typename T>

void deletion(Node<T>\* head){

cout<<"\n ==============================================="<<endl;

//delete first element

Node<T>\* temp=head;

Node<T>\* last=head;

while(last->next!=head){

last=last->next;

}

head=head->next;

last->next=head;

delete temp;

cout<<"After deleting first element: ";

Traversal(head);

cout<<"\n ============================================"<<endl;

//delete middle element

int element;

cout<<"Enter the element you wanna delete: ";

cin>>element;

Node<T>\* temp2=head;

while(temp2->next!=head && temp2->next->data!=element){

temp2=temp2->next;

}

if(temp2->next!=head){

Node<T>\* ele1=temp2->next;

temp2->next=ele1->next;

delete ele1;

}

cout<<"\n After deleting given element: ";

Traversal(head);

cout<<"====================================================="<<endl;

//deleting last element

Node<T>\* temp3=head;

while(temp3->next->next!=head){

temp3=temp3->next;

}

Node<T>\* last3=temp3->next;

temp3->next=head;

delete last3;

cout<<"\n After deleting last element: ";

Traversal(head);

cout<<"\n ========================================================"<<endl;

}

//------------------------------------------------------------------------------

template <typename T>

void looping(Node<T>\* head){

int size=0;

cout<<"Enter how much elements of list you wanna see by looping: ";

cin>>size;

Node<T>\* temp=head;

for(int i=0;i<size;i++){

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

temp=temp->next;

}

cout<<"\n=============================================="<<endl;

}

int main(){

Node<int>\* head=nullptr;

head=new Node<int>(1);

Node<int>\* second=new Node<int>(2);

Node<int>\* third=new Node<int>(4);

Node<int>\* forth=new Node<int>(5);

Node<int>\* fifth=new Node<int>(6);

head->next=second;

second->next=third;

third->next=forth;

forth->next=fifth;

fifth->next=head;

Traversal(head);

int choice=6;

cout<<"====================================================";

while(choice!=0){

cout<<"Enter your choice: "<<endl;

cout<<"1)Insertion \n2)Deletion \n3)Looping of list\n0)End programme"<<endl;

cout<<"\n Choice: ";

cin>>choice;

switch (choice){

case(0):{

return 0;

}

case(1):{

insertion(head);

break;

}

case(2):{

deletion(head);

break;

}

case(3):{

looping(head);

break;

}

default:{

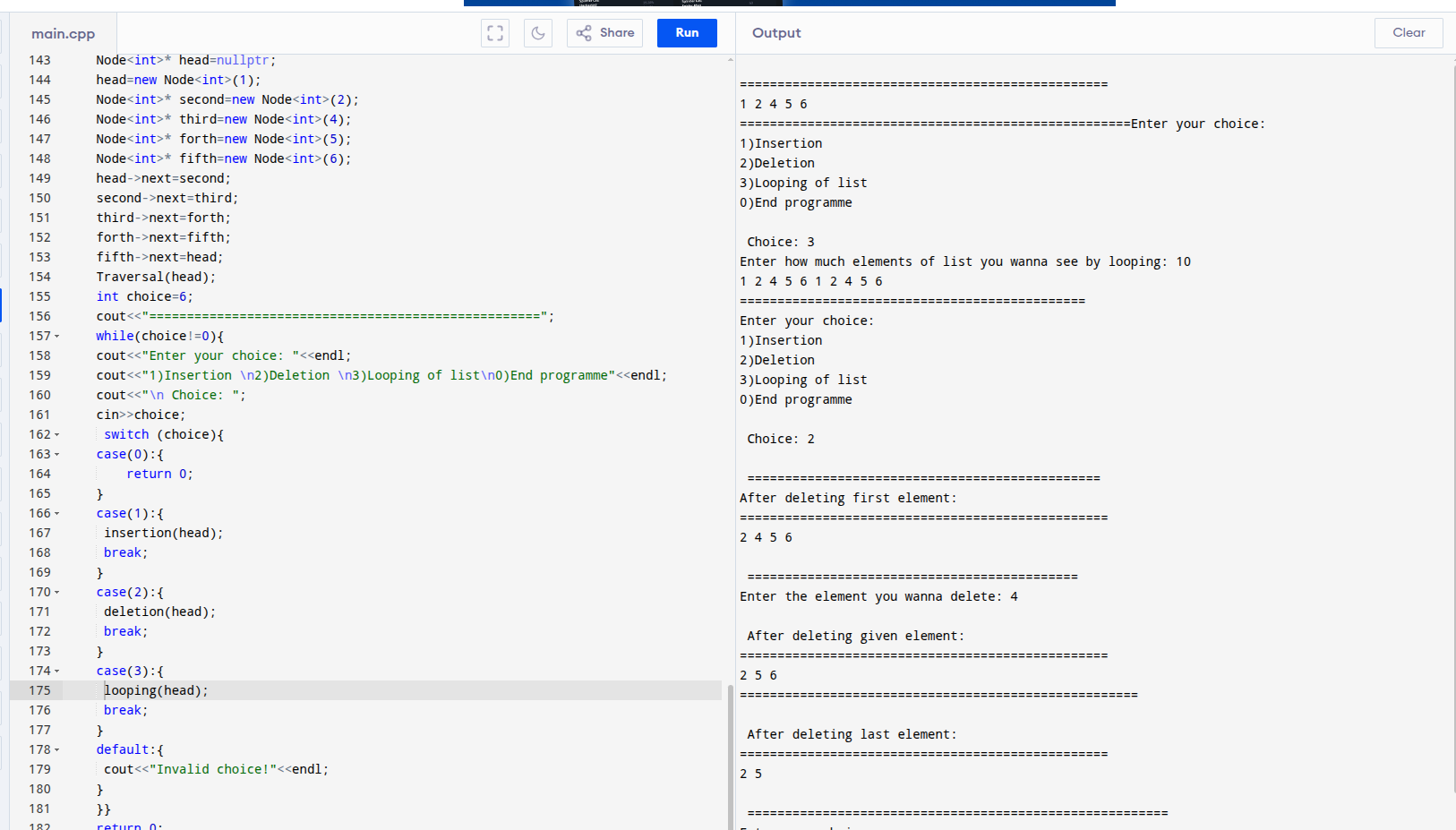
cout<<"Invalid choice!"<<endl;

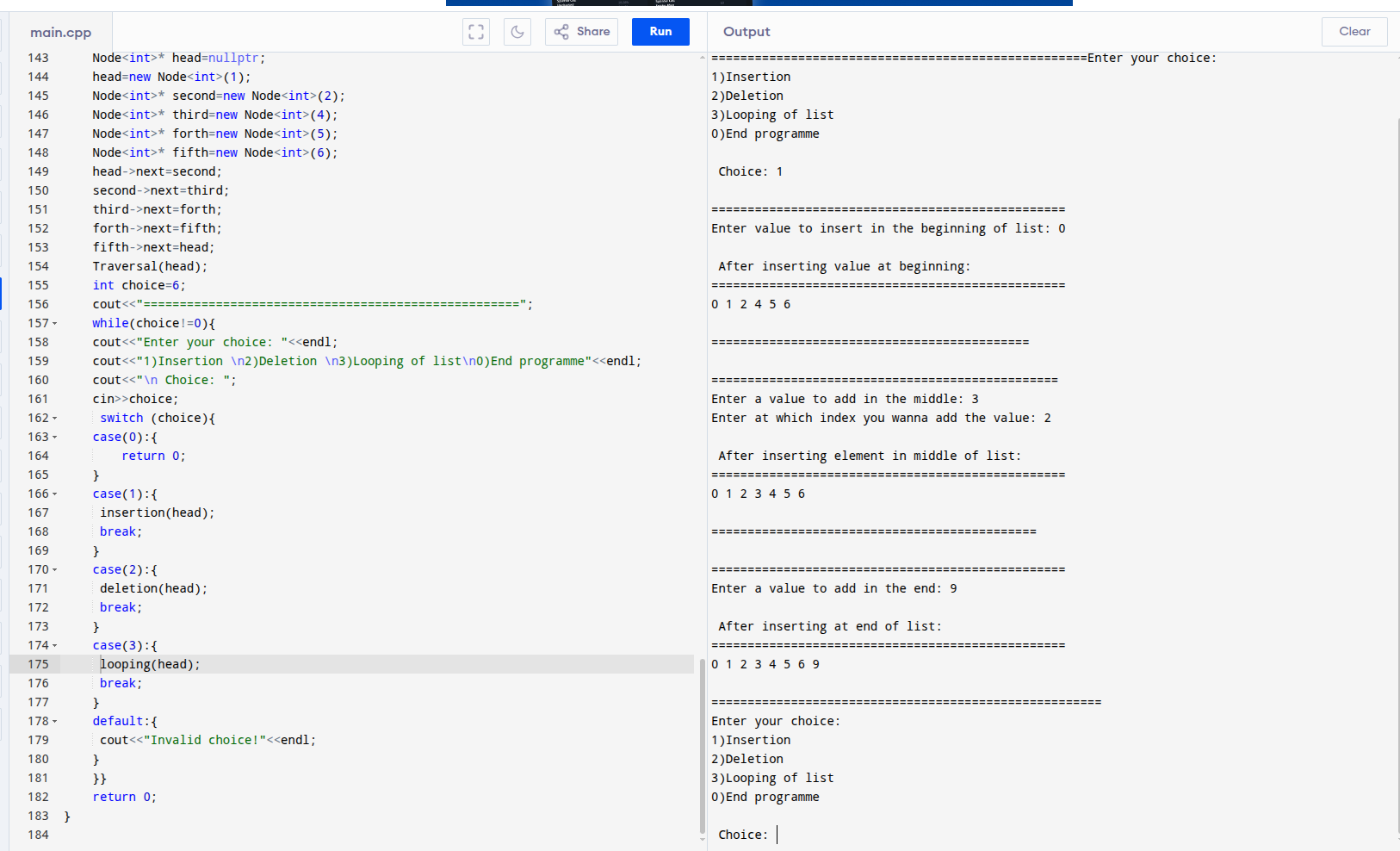
}

}}

return 0;

}





**Task(Weekly):**

#include<iostream>

using namespace std;

template<typename T>

class Node{

public:

T data;

Node<T>\* prev;

Node<T>\* next;

Node(T value){

this->data= value;

this->prev=nullptr;

this->next=nullptr;

}

};

//-----------------------------------------------------------------

template<typename T>

void Traversal(Node<T>\* &head){

if(!head){

cout<<"this list is empty";

cout<<"\n ======================================================"<<endl;

}

//forward traversal

cout<<"Forward Traversal: ";

Node<T>\* temp1=head;

do{

cout<<temp1->data<<" ";

temp1=temp1->next;

}while(temp1!=head);

cout<<"\n =================================================="<<endl;

//backword traversal

temp1=head->prev;

Node<T>\* last\_node1=temp1;

cout<<"\n Backword Traversal: ";

do{

cout<<temp1->data<<" ";

temp1=temp1->prev;

}while(temp1!=last\_node1);

}

//---------------------------------------------------------------------------

template <typename T>

void insertion(Node<T>\* head){

cout<<"\n =======================================================";

//insertion at beginning

T value;

cout<<"\n Enter a value to add in the beginning: ";

cin>>value;

Node<T>\* new\_node1=new Node<T>(value);

Node<T>\* temp1=head;

if(head==nullptr){

head=new\_node1;

head->next=head;

head->prev=head;

}

else{

Node<T>\* last1 = head->prev;

new\_node1->next=head;

new\_node1->prev = last1;

head->prev=new\_node1;

last1->next=new\_node1;

head=new\_node1;

}

cout<<"\n After inserting one value at the beginning: \n";

Traversal(head);

cout<<"\n ====================================================================";

//insertion in middle

T value2, index;

int count=0;

cout<<"\n Enter value to insert in the middle of list: ";

cin>>value2;

cout<<"Enter at which index you wanna add the value: ";

cin>>index;

Node<T>\* temp2=head;

do{

count++;

temp2=temp2->next;

}while(temp2!=head && count!=index);

Node<T>\* new\_node2=new Node<T>(value2);

if(count==index){

new\_node2->next=temp2;

new\_node2->prev=temp2->prev;

temp2->prev->next=new\_node2;

temp2->prev=new\_node2;

}

cout<<"\n After traversing in middle: \n";

Traversal(head);

cout<<"\n======================================================"<<endl;

//insertion at end

T value3;

cout<<"\n Enter a value to add in the end of the list: ";

cin>>value3;

Node<T>\* last3=head->prev;

Node<T>\* new\_node3=new Node<T>(value3);

new\_node3->next=head;

new\_node3->prev=last3;

head->prev=new\_node3;

last3->next=new\_node3;

cout<<"\n After inserting element in end: \n";

Traversal(head);

cout<<"\n ================================================================"<<endl;

}

//--------------------------------------------------------------------------

template <typename T>

void deletion(Node<T>\* &head){

cout<<"\n==========================================================";

//deletion in beginning

if(head==nullptr){

cout<<"List is empty!"<<endl;

}

else{

Node<T>\* temp1=head;

head->next->prev=head->prev;

head->prev->next=head->next;

head=head->next;

delete temp1;

}

cout<<"\n After deleting 1st value: \n";

Traversal(head);

cout<<"\n ================================================="<<endl;

//deleting middle vaue

T index,count2=0;

cout<<"\n Enter at which index you wanna delete: ";

cin>>index;

index--;

Node<T>\* temp2=head;

do{

count2++;

temp2=temp2->next;

}while(count2!= index);

if(count2==index){

temp2->prev->next=temp2->next;

temp2->next->prev=temp2->prev;

delete temp2;

}

cout<<"\n After deleting middle value: "<<endl;

Traversal(head);

cout<<"\n ========================================================"<<endl;

//delete from end

Node<T>\* last3=head->prev;

last3->prev->next=last3->next;

head->prev=last3->prev;

delete last3;

cout<<"\n After deleting last value: \n";

Traversal(head);

cout<<"\n ===================================";

}

//------------------------------------------------------------------------------

int main(){

Node<int>\* head=nullptr;

head=new Node<int>(1);

Node<int>\* second=new Node<int>(2);

Node<int>\* third=new Node<int>(4);

Node<int>\* forth=new Node<int>(5);

Node<int>\* fifth=new Node<int>(6);

head->next=second;

second->next=third;

third->next=forth;

forth->next=fifth;

fifth->next=head;

fifth->prev=forth;

forth->prev=third;

third->prev=second;

second->prev=head;

head->prev=fifth;

Traversal(head);

int choice;

cout<<"\n ====================================================";

while(choice!=0){

cout<<"\n Enter your choice: "<<endl;

cout<<"1)Insertion \n2)Deletion \n3)Traversal\n0)End programme"<<endl;

cout<<"\n Choice: ";

cin>>choice;

switch (choice){

case(1):{

insertion(head);

break;

}

case(2):{

deletion(head);

break;

}

case(3):{

Traversal(head);

break;

}

default:{

cout<<"Invalid choice!"<<endl;

}

}

return 0;

}

}

