DATA STRUCTURES

Division	CS(AIML) -A
Batch	2
GR-no	12311493
Roll no	54
Name	Atharva Kangralkar

Assignment 5:

Write a program to implement Stack and Queue basic operations

Code:-

1)Stack:-

```
#include<iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
    Node(int val) {
        data = val;
        next = nullptr;
    }
};
class Stack {
public:
    Node* top;
public:
    Stack() {
        top = nullptr;
    }
    bool isEmpty() {
        return top == nullptr;
    }
    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
        cout << "Pushed to stack : "<< x;
        cout << "\n";
    }
    void pop() {
        if(isEmpty()) {
            cout << "Stack underflow! cannot pop";
            return;
        }
        Node* temp = top;
        cout << "Popped : " << top->data;
        top = top->next;
        cout << "\n";
        delete temp;
    }
    int peek() {
        if(isEmpty()) {
            cout<<"Stack is empty!";
            return -1;
        }
        return top->data;
    }
    void display() {
        if(isEmpty()) {
            cout<<"Stack is empty!";
            return;
        }
        Node* temp = top;
        cout << "Stack elements : ";
        while(temp) {
            cout << temp->data << " -> ";
            temp = temp->next;
        }
        cout << "NULL \n";
    }
};
int main() {
    Stack s;
    s.push(5);
    s.push(10);
    s.push(15);
    s.display();
    cout << "Top element : " << s.peek();
    cout << "\n";
    s.pop();
    s.display();
}
```

2) Queue:-

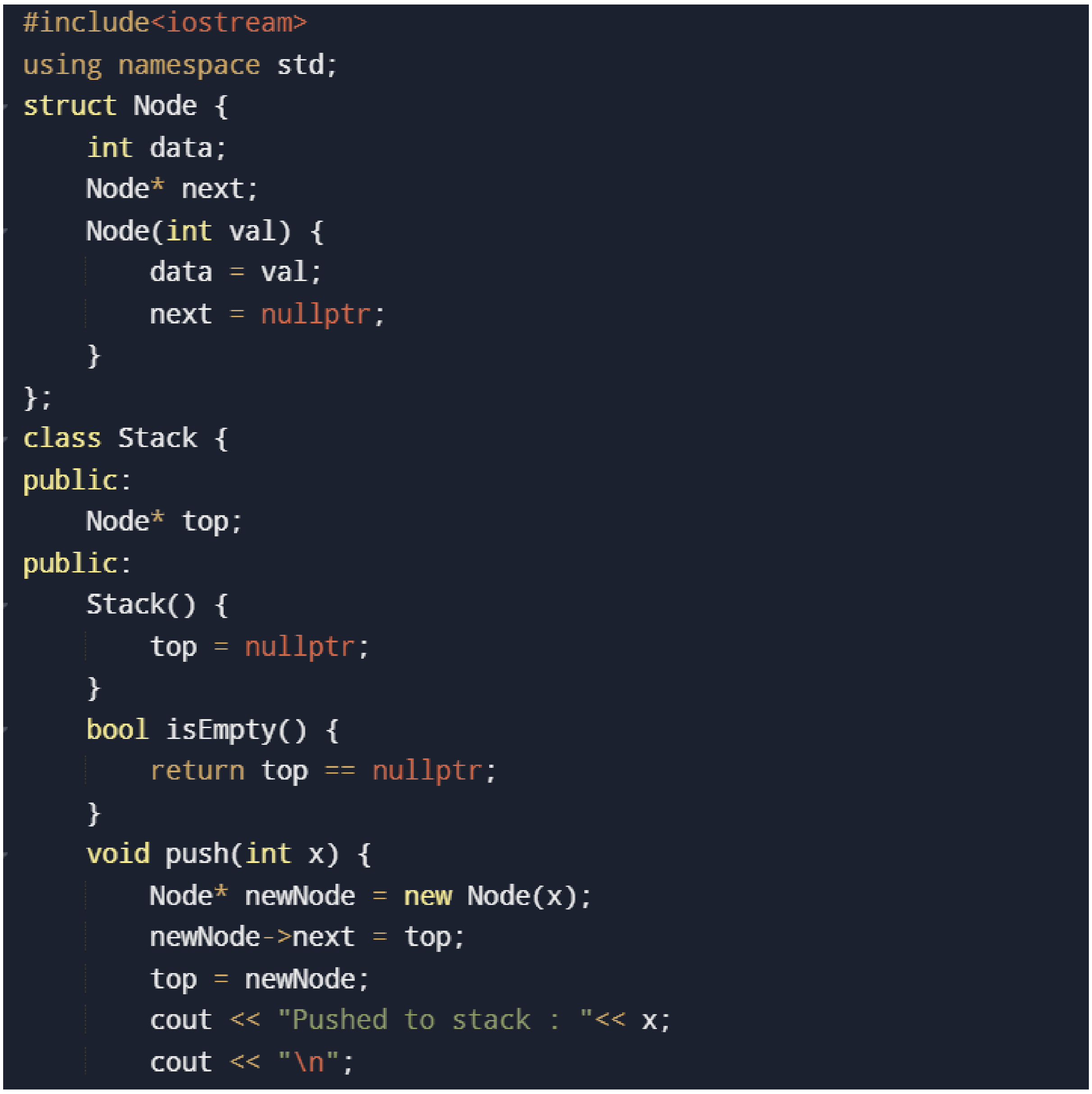
```
#include <iostream>

using namespace std;

struct Node {

 int data;
 Node* next;
 Node(int val) {
 data = val;
 next = nullptr;
 }
};
class Queue {
public:
 Node* front;
 Node* rear;
public:
 Queue() {
 front = nullptr;
 rear = nullptr;
 }
 bool isEmpty() {
 return front == nullptr;
 }
 void enqueue(int val) {
 Node* newNode = new Node(val);
 if(rear == nullptr) {
 front = rear = newNode;
 }
 else {
 rear->next = newNode;
 rear = newNode;
 }
 }
 void dequeue() {
```

```
if(isEmpty()) {
cout<<"Queue is empty!\n";
return ;
}
Node* temp = front;
front = front->next;
if(front == nullptr) {
rear = nullptr;
}
delete temp;
}
int peek() {
if(isEmpty()) {
cout << "Queue is empty!";
return -1;
}
return front->data;
cout<<"\n";
}
void display() {
if(isEmpty()) {
cout << "Queue is empty!";
return;
}
Node* temp = front;
while(temp) {
cout << temp->data << " -> ";
temp = temp->next;
}
```

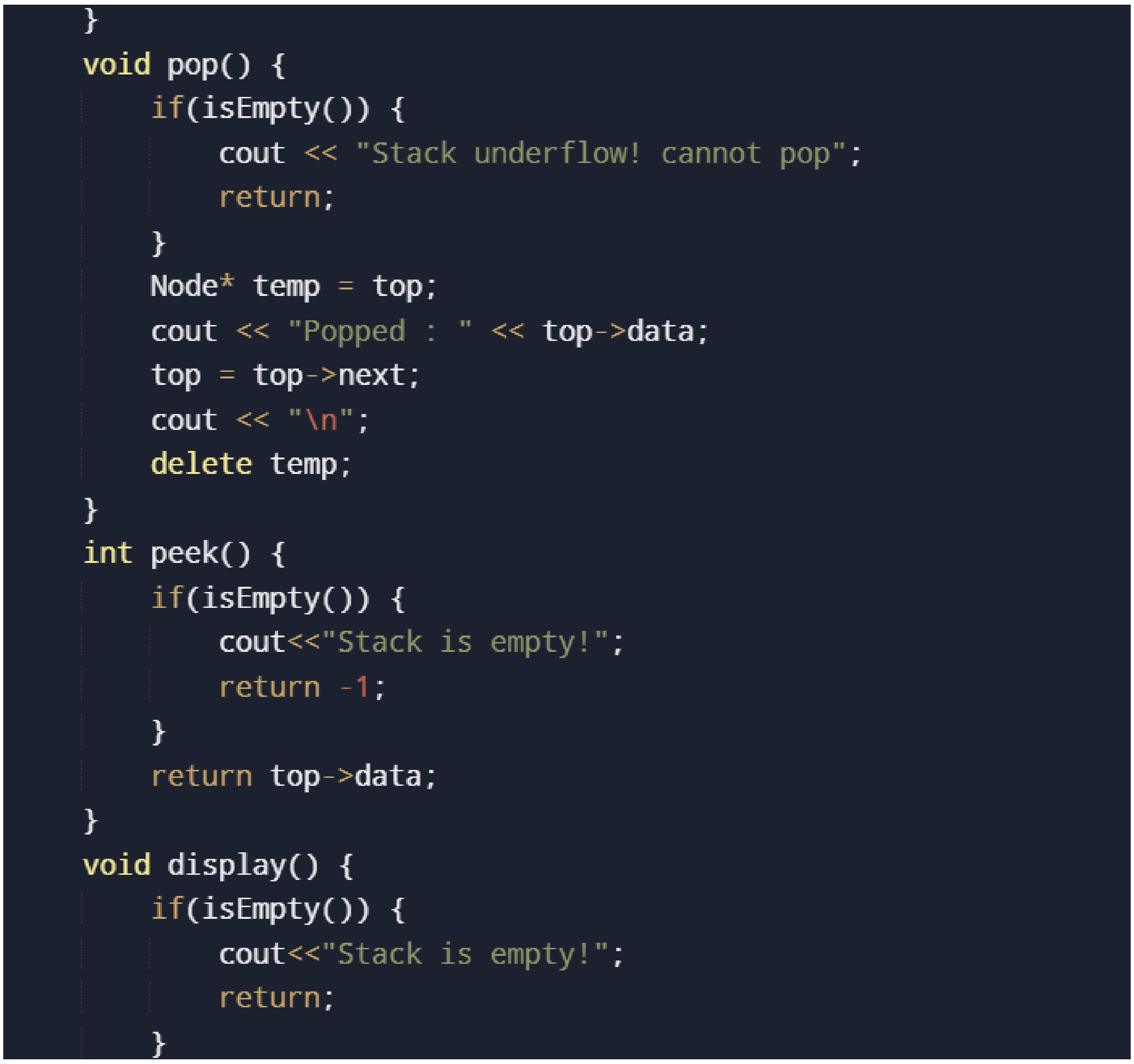
```
 cout << "NULL \n";
 }
};
int main() {
 Queue q;
 q.enqueue(10);
 q.enqueue(20);
 q.enqueue(30);
 q.enqueue(40);
 q.display();
 cout<<"Front element : " << q.peek() << endl;
 q.dequeue();
 q.dequeue();
 q.dequeue();
 q.display();
 return 0;
}
```

Code Screenshot:-

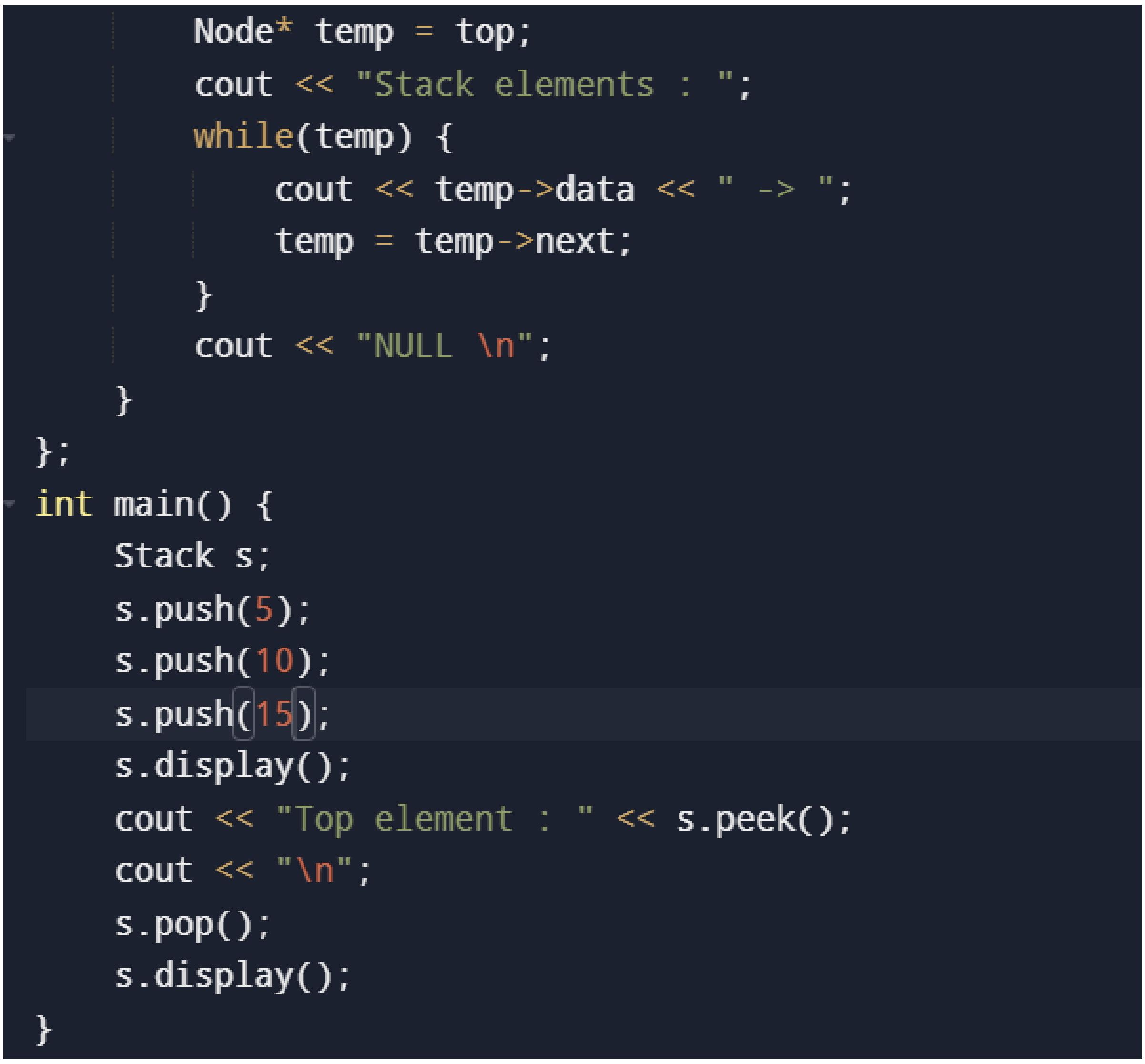
1)Stack



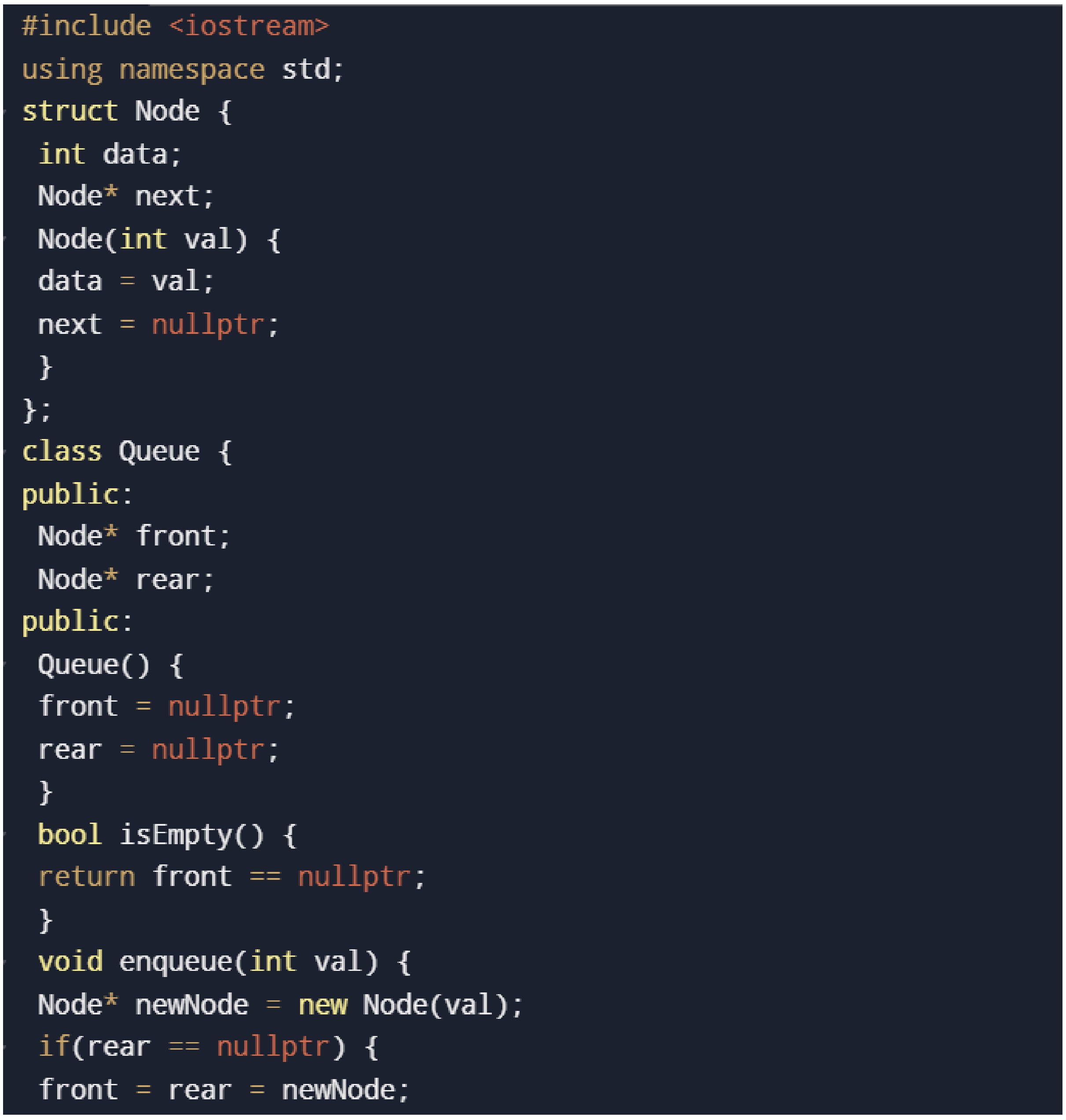
```cpp
#include<iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
    Node(int val) {
        data = val;
        next = nullptr;
    }
};
class Stack {
public:
    Node* top;
public:
    Stack() {
        top = nullptr;
    }
    bool isEmpty() {
        return top == nullptr;
    }
    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
        cout << "Pushed to stack : "<< x;
        cout << "\n";
```



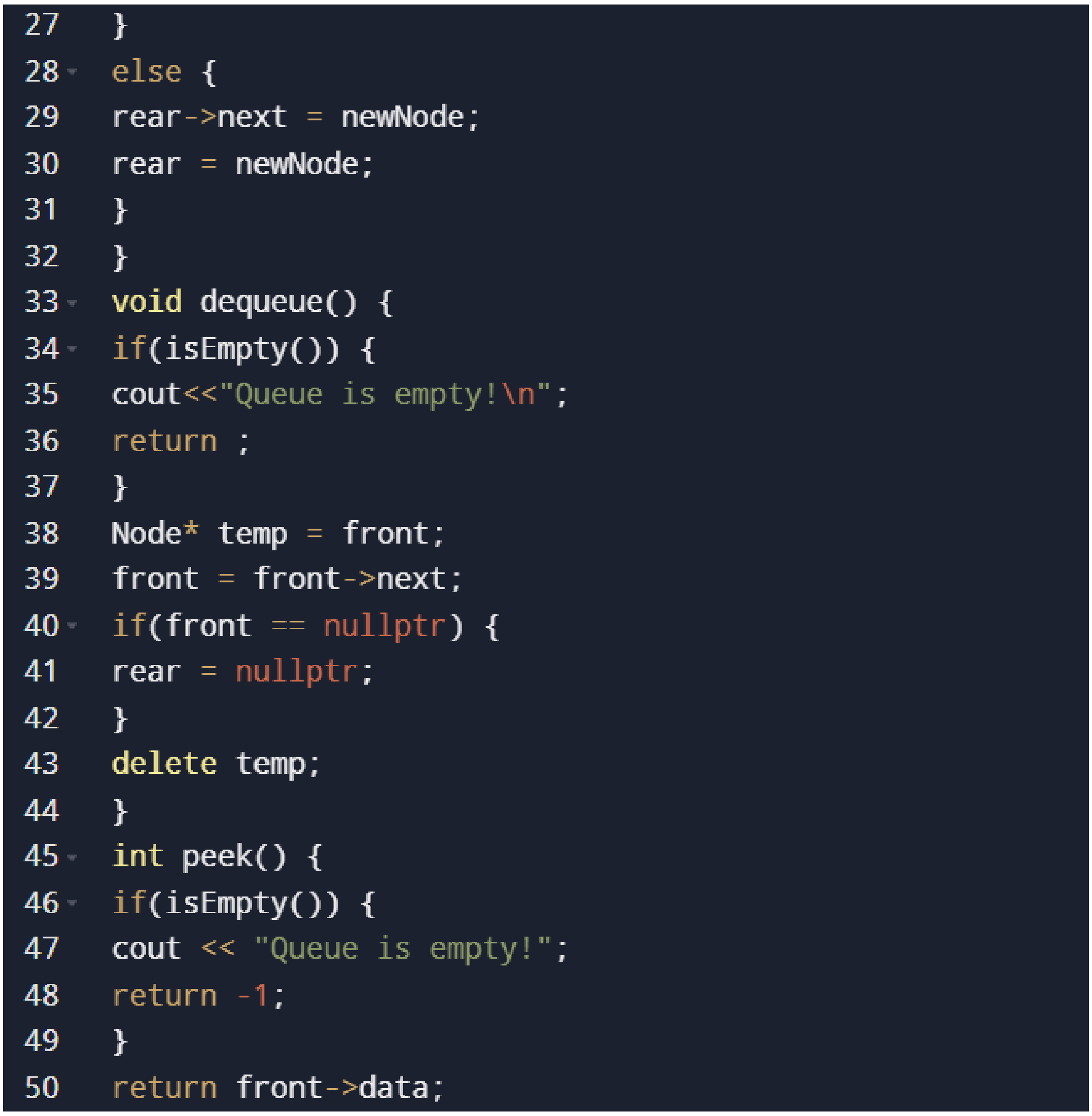
```cpp
    }
    void pop() {
        if(isEmpty()) {
            cout << "Stack underflow! cannot pop";
            return;
        }
        Node* temp = top;
        cout << "Popped : " << top->data;
        top = top->next;
        cout << "\n";
        delete temp;
    }
    int peek() {
        if(isEmpty()) {
            cout<<"Stack is empty!";
            return -1;
        }
        return top->data;
    }
    void display() {
        if(isEmpty()) {
            cout<<"Stack is empty!";
            return;
        }
```


```
        Node* temp = top;
        cout << "Stack elements : ";
        while(temp) {
            cout << temp->data << " -> ";
            temp = temp->next;
        }
        cout << "NULL \n";
    }
};
int main() {
    Stack s;
    s.push(5);
    s.push(10);
    s.push(15);
    s.display();
    cout << "Top element : " << s.peek();
    cout << "\n";
    s.pop();
    s.display();
}
```

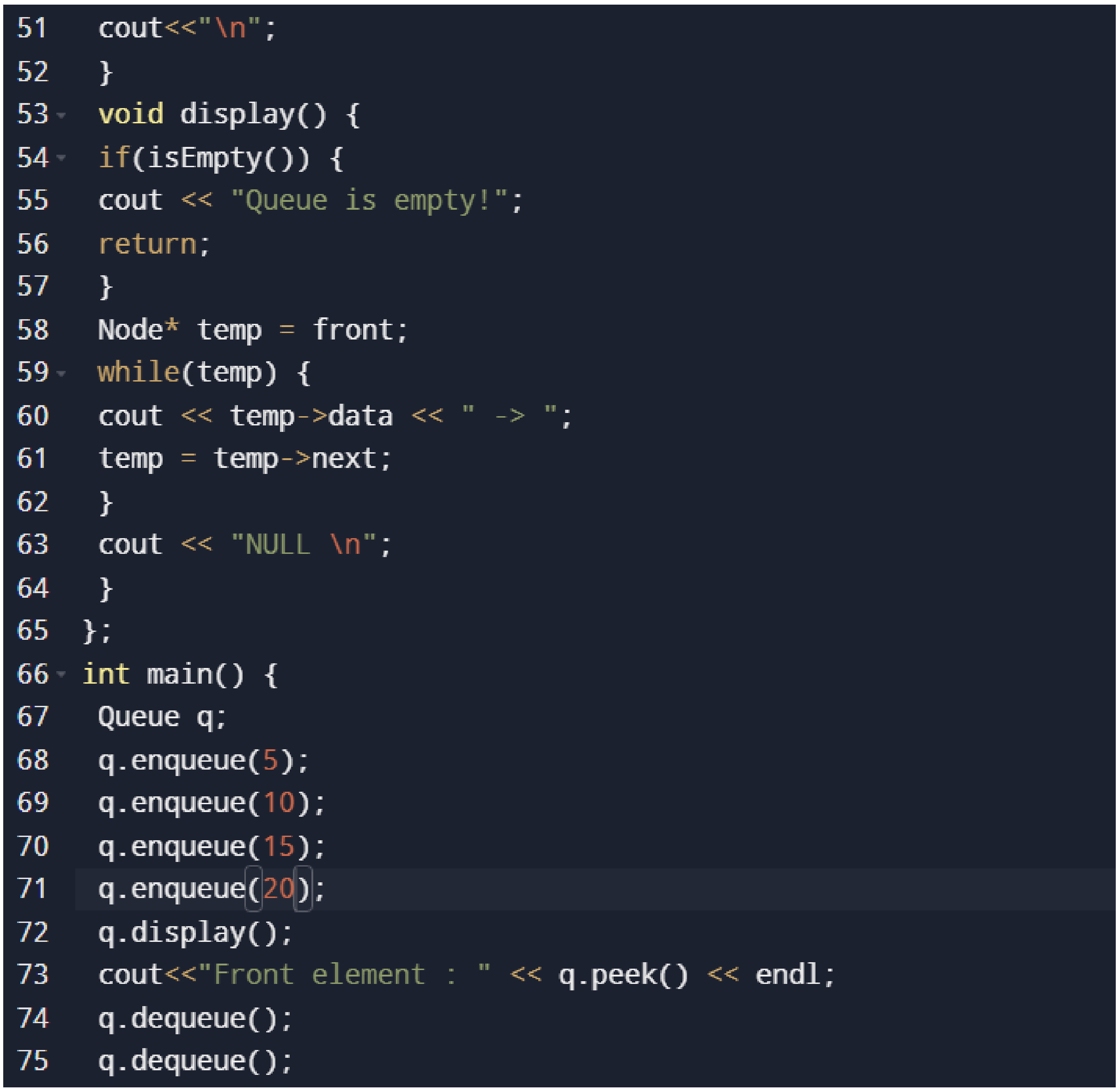
2)Queue:-



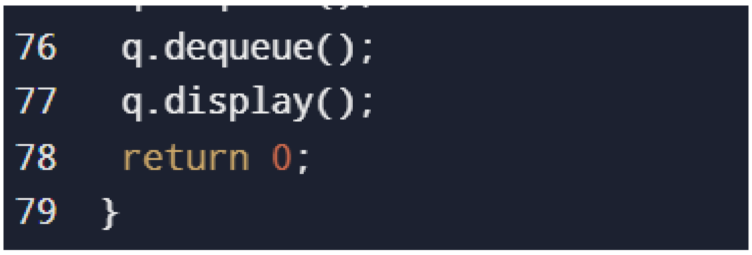
```
#include <iostream>
using namespace std;
struct Node {
 int data;
 Node* next;
 Node(int val) {
 data = val;
 next = nullptr;
 }
};
class Queue {
public:
 Node* front;
 Node* rear;
public:
 Queue() {
 front = nullptr;
 rear = nullptr;
 }
 bool isEmpty() {
 return front == nullptr;
 }
 void enqueue(int val) {
 Node* newNode = new Node(val);
 if(rear == nullptr) {
 front = rear = newNode;
```



```cpp
}
else {
rear->next = newNode;
rear = newNode;
}
}
void dequeue() {
if(isEmpty()) {
cout<<"Queue is empty!\n";
return ;
}
Node* temp = front;
front = front->next;
if(front == nullptr) {
rear = nullptr;
}
delete temp;
}
int peek() {
if(isEmpty()) {
cout << "Queue is empty!";
return -1;
}
return front->data;
```



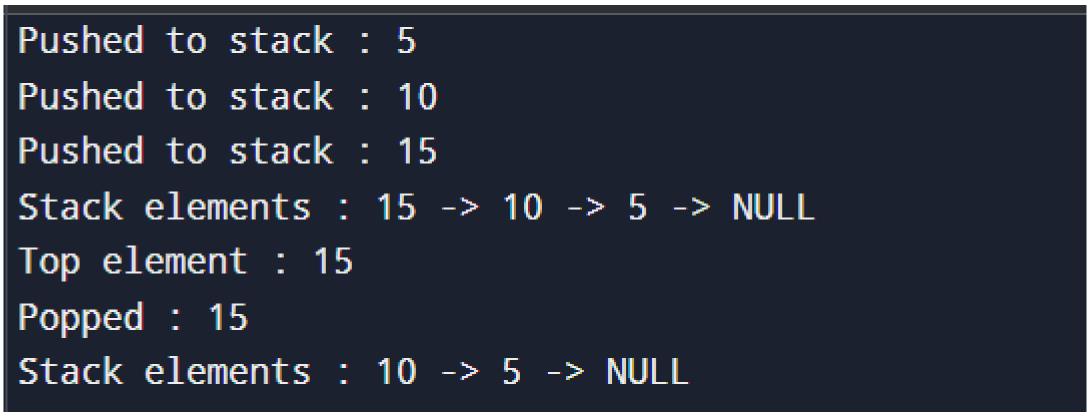
```cpp
 cout<<"\n";
 }
 void display() {
 if(isEmpty()) {
 cout << "Queue is empty!";
 return;
 }
 Node* temp = front;
 while(temp) {
 cout << temp->data << " -> ";
 temp = temp->next;
 }
 cout << "NULL \n";
 }
};
int main() {
 Queue q;
 q.enqueue(5);
 q.enqueue(10);
 q.enqueue(15);
 q.enqueue(20);
 q.display();
 cout<<"Front element : " << q.peek() << endl;
 q.dequeue();
 q.dequeue();
```



```cpp
 q.dequeue();
 q.display();
 return 0;
}
```

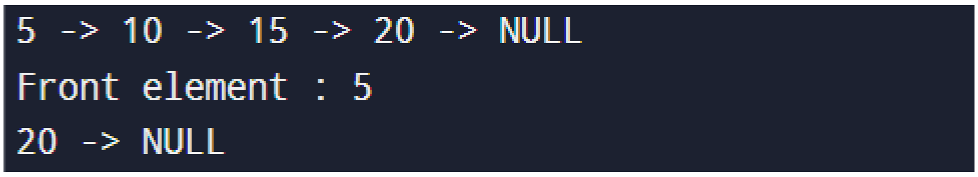
Output:-

1)Stack



```
Pushed to stack : 5
Pushed to stack : 10
Pushed to stack : 15
Stack elements : 15 -> 10 -> 5 -> NULL
Top element : 15
Popped : 15
Stack elements : 10 -> 5 -> NULL
```

2)Queue



```
5 -> 10 -> 15 -> 20 -> NULL
Front element : 5
20 -> NULL
```