LAB REPORT:

Group members:

- Hira Arif
- Laiba Batool
- Mariyam Muzammil

Course:	DSA
Lab Number:	03
Lab Title:	Stack and Queue Implementation
Date:	16/10/24

Objectives

1. Practice Abstract Data Types.

2. Understand the Basic of Stack and Queue.

3. Implement Stack and Queue using Linked List and Arrays.

Conclusion:

The main function is working well. All functions are good to go. We've become fully aware of how to implement Stack and Queue using Arrays and Linked list.

QUEUE (Circular Queue):

IMPLEMENTATION BY ARRAYS:

Header file (.h file):

```
#ifndef QUEUE_H
#define QUEUE_H


class Queue
{
    public:
        Queue(int size);
        virtual ~Queue();

        bool isEmpty();
        bool isFull();
        void Enqueue(double x);
        double Dequeue();
        void print();

    protected:

    private:
        int size;
        int f;
        int r;
        double *array;
};

#endif // QUEUE_H
```

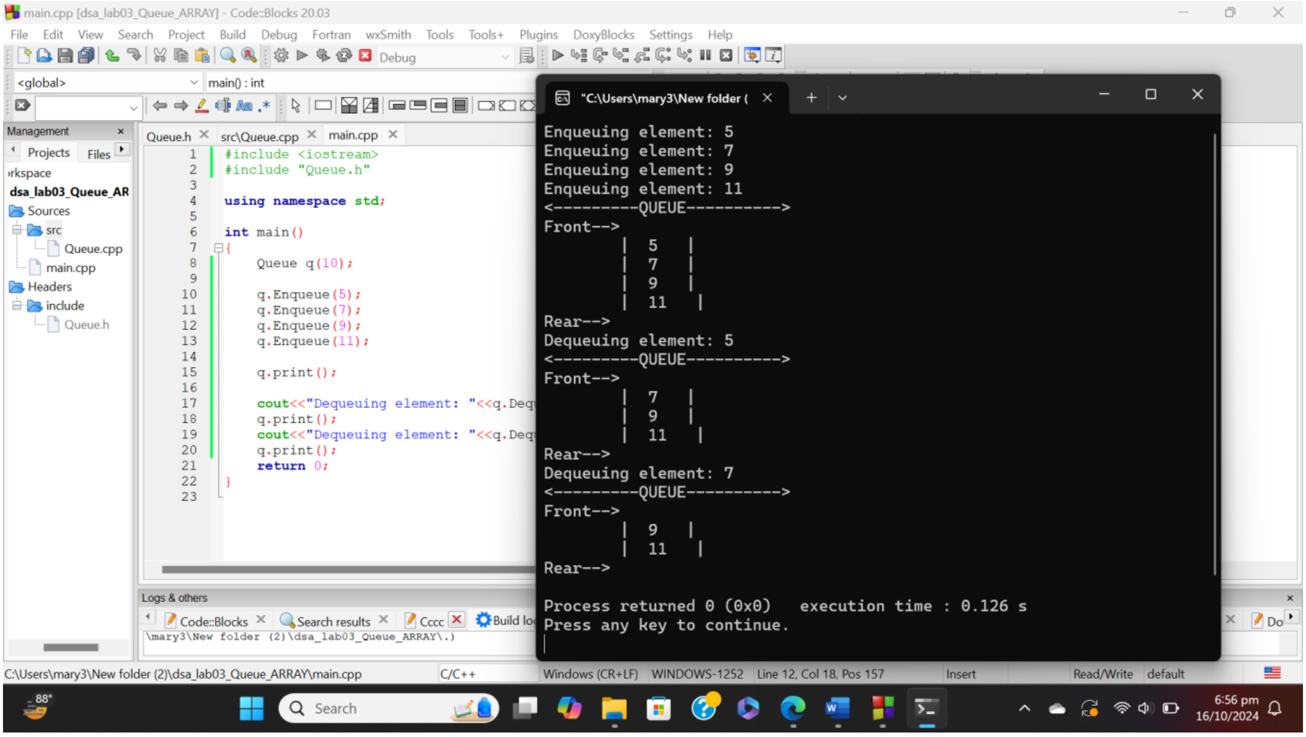
Implementation File (.CPP):

```
#include "Queue.h"
#include <iostream>
using namespace std;

Queue::Queue(int size)
{
    array= new double[size];
    f=r=0;
}

Queue::~Queue()
{
   delete [] array;
}



bool Queue::isEmpty(){
  if(f==r){
    return true;
  }else
    return false;
}

bool Queue::isFull(){
  if((r+1)%size==f){
    return true;
  }else
    return false;
}



void Queue::Enqueue(double x){
   if(isFull()){
    cout<<"Can't insert element as queue is full"<<endl;
   }else{
    r=(r+1)%size;
    array[r]=x;
    cout<<"Enqueuing element: "<<x<<endl;
   }
}

double Queue::Dequeue(){
   if(isEmpty()){
    cout<<"Can't remove element as queue is empty"<<endl;
    return -1;
   }else{
       f=(f+1)%size;
       return array[f];
   }
}
void Queue::print(){
   cout<<"<---------QUEUE---------->"<<endl;
   cout<<"Front-->"<<endl;
   int i=(f+1)%size;
   while(i!=(r+1)%size){
    cout<<"        |  "<<array[i]<<"   |"<<endl;
    i=(i+1)%size;
   }
   cout<<"Rear-->"<<endl;
}
```

Main File:

```
#include <iostream>
#include "Queue.h"

using namespace std;

int main()
{
    Queue q(10);

    q.Enqueue(5);
    q.Enqueue(7);
    q.Enqueue(9);
    q.Enqueue(11);

    q.print();

    cout<<"Dequeuing element: "<<q.Dequeue()<<endl;
    q.print();
    cout<<"Dequeuing element: "<<q.Dequeue()<<endl;
    q.print();
    return 0;
}
```

DIRECTORY:



```
#include <iostream>
#include "Queue.h"

using namespace std;

int main()
{
    Queue q(10);

    q.Enqueue(5);
    q.Enqueue(7);
    q.Enqueue(9);
    q.Enqueue(11);

    q.print();

    cout<<"Dequeuing element: "<<q.Deq
    q.print();
    cout<<"Dequeuing element: "<<q.Deq
    q.print();
    return 0;
}
```

```
Enqueuing element: 5
Enqueuing element: 7
Enqueuing element: 9
Enqueuing element: 11
<---------QUEUE---------->
Front-->
         |  5   |
         |  7   |
         |  9   |
         |  11   |
Rear-->
Dequeuing element: 5
<---------QUEUE---------->
Front-->
         |  7   |
         |  9   |
         |  11   |
Rear-->
Dequeuing element: 7
<---------QUEUE---------->
Front-->
         |  9   |
         |  11   |
Rear-->

Process returned 0 (0x0)   execution time : 0.126 s
Press any key to continue.
```

IMPLEMENTATION BY LINKED LIST:

Header file (.h file):

```
#ifndef QUEUE_H
#define QUEUE_H


struct Node{
   double data;
   Node* next;
};
class Queue
{
    public:
        Queue();
        virtual ~Queue();
        bool isEmpty();
        void Enqueue(double x);
        double Dequeue();
        void print();


    protected:

    private:
        Node* front;
        Node* rear;
        int counter;
};

#endif // QUEUE_H
```

Implementation File (.CPP):

```
#include "Queue.h"
#include<iostream>
using namespace std;

Queue::Queue()
{
    front=rear=NULL; //setting to NULL as queue is empty
    counter=0;
}

Queue::~Queue()
{
    while(!isEmpty()){
    Dequeue();}
}


bool Queue::isEmpty(){
    if(front==NULL){
       return true;
    }else
       return false;
 }



void Queue::Enqueue(double x){
    Node* ptr=new Node();
    if(ptr==NULL)
   {
        cout<<"Cant enqueue as queue is full"<<endl;
   } else
   {

    ptr->data=x;
    ptr->next=NULL;
    if(isEmpty())
    {
        front=ptr;  //pointing both front and rear to the added node
    }else
    {
        rear->next=ptr;
    }
```

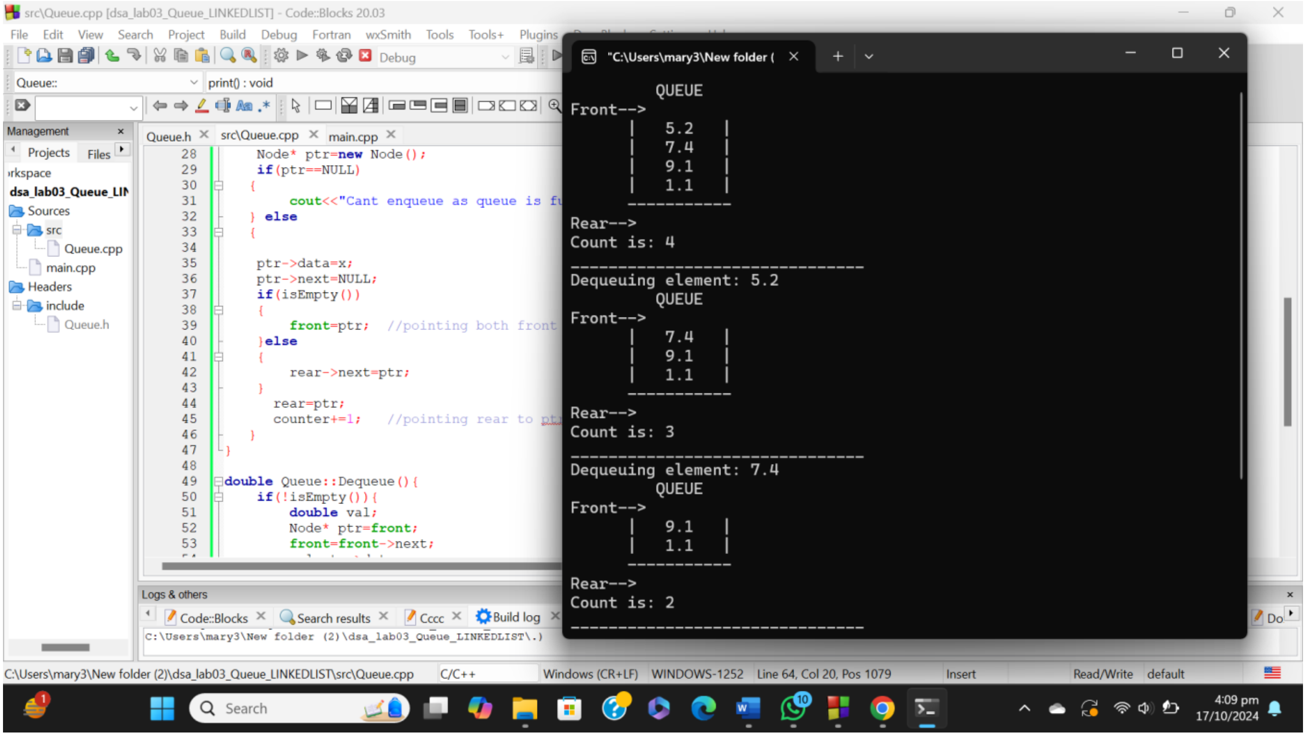
```
      rear=ptr;
      counter+=1;   //pointing rear to ptr in both cases
   }
}

double Queue::Dequeue(){
    if(!isEmpty()){
        double val;
        Node* ptr=front; //making temp ptr to delete front data
        front=front->next;
        val=ptr->data;
        delete ptr;
        counter-=1;
        return val;
    }else{
        cout<<"Cant dequeue as queue is empty"<<endl;
        return -1;
    }
}
void Queue::print(){
    cout<<"        QUEUE"<<endl;
    cout<<"Front-->"<<endl;
    Node* temp=front;
    while(temp!=NULL){
        cout<<"      |    "<<temp->data<<"    |"<<endl;
        temp=temp->next;
    }
    cout<<"      -----------"<<endl;
    cout<<"Rear-->"<<endl;
    cout<<"Count is: "<<counter<<endl;
    cout<<"________________________________________"<<endl;
```

```
}
```

Main File:

```
#include <iostream>
#include "Queue.h"

using namespace std;

int main()
{
    Queue q;

    q.Enqueue(5.2);
    q.Enqueue(7.4);
    q.Enqueue(9.1);
    q.Enqueue(1.1);

    q.print();

    cout<<"Dequeuing element: "<<q.Dequeue()<<endl;
    q.print();
    cout<<"Dequeuing element: "<<q.Dequeue()<<endl;
    q.print();

    return 0;
}
```

DIRECTORY:



```
    Node* ptr=new Node();
    if(ptr==NULL)
    {
        cout<<"Cant enqueue as queue is fu
    } else
    {

    ptr->data=x;
    ptr->next=NULL;
    if(isEmpty())
    {
        front=ptr;  //pointing both front
    }else
    {
        rear->next=ptr;
    }
      rear=ptr;
      counter+=1;   //pointing rear to ptr
    }
}

double Queue::Dequeue(){
    if(!isEmpty()){
        double val;
        Node* ptr=front;
        front=front->next;
```

```
C:\Users\mary3\New folder (2)\dsa_lab03_Queue_LINKEDLIST\.)
```

```
          QUEUE
Front-->
       |   5.2   |
       |   7.4   |
       |   9.1   |
       |   1.1   |
       -----------
Rear-->
Count is: 4

-------------------------------
Dequeuing element: 5.2
          QUEUE
Front-->
       |   7.4   |
       |   9.1   |
       |   1.1   |
       -----------
Rear-->
Count is: 3

-------------------------------
Dequeuing element: 7.4
          QUEUE
Front-->
       |   9.1   |
       |   1.1   |
       -----------
Rear-->
Count is: 2

-------------------------------
```

STACK:

IMPLEMENTATION BY ARRAYS:

Header File(.h File):

```
#ifndef STACK_H

#define STACK_H


class Stack{

    public:

      Stack();

      Stack(int size);

      ~Stack();


      bool empty() const;

      void push(const double x);

      double pop();
```

```
        bool full();
        double Top() const;
        void print() const;

    private:
        int top; //index of top element of stack
        int maxTop;
        double* values;
};
#endif
```

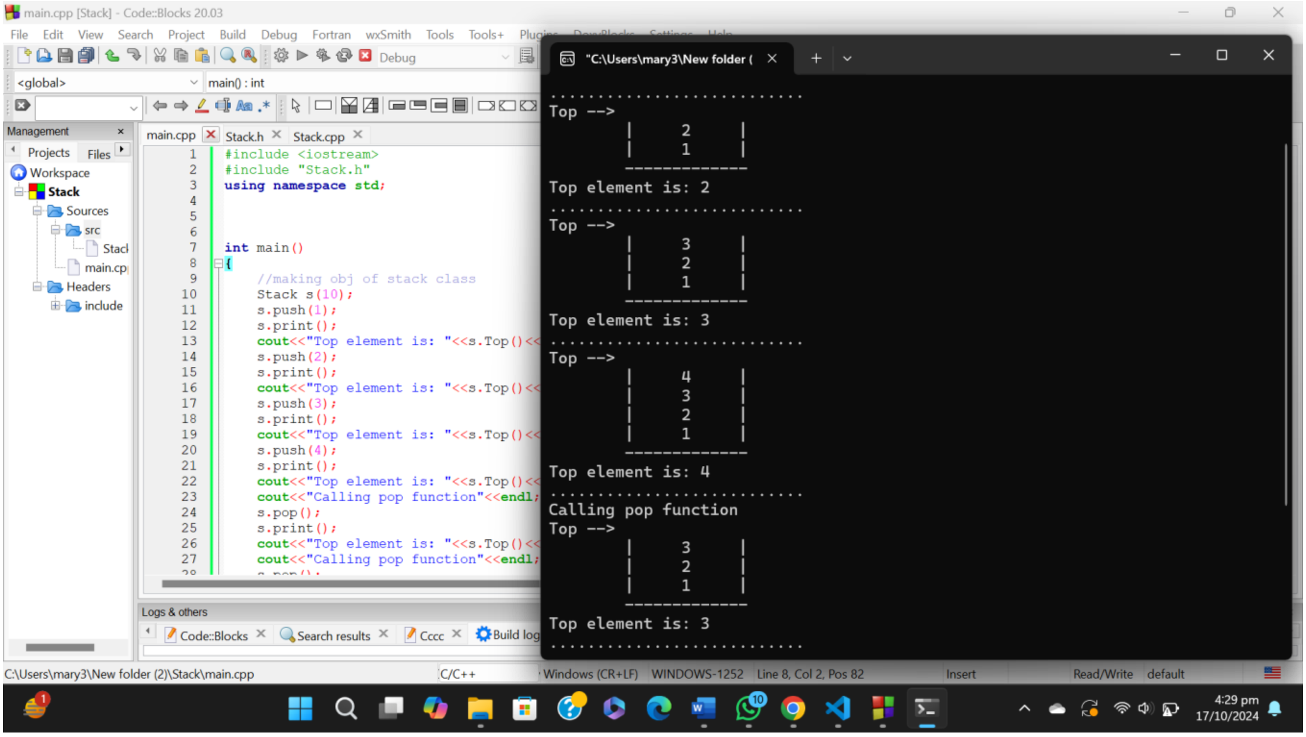
Implementation File (.CPP File):

```
#include "Stack.h"
#include <iostream>
using namespace std;

Stack::Stack(){}
Stack::Stack(int size){
    values= new double[size];
    maxTop=size-1;  //max index
    top=-1;  //initially, the stack is empty
}

Stack::~Stack(){
    delete [] values;  //free the dynamically allocated array
}

bool Stack::full(){
    if(top==maxTop){
        cout<<"Stack is full"<<endl;
        return true;
    }else
        return false;
}

bool Stack::empty() const{
    if(top==-1){
        cout << "Stack is empty"<< endl;
        return true;
    } else
        return false;
}

void Stack::push(const double x){
    if(full()){
        cout << "Stack is full"<< endl;
    }else{
        top+=1;
     values[top]=x;
    }

}
double Stack::pop(){
    if(empty()){
        cout << "Stack is empty"<< endl;
        return -1;
    }else{
        top-=1;
        return values[top];
    }
}
double Stack::Top() const{
    if(empty()){
        cout << "Stack is empty"<< endl;
        return -1;
    }else{
        return values[top];
    }
}
void Stack::print() const{
    cout<<"Top --> "<<endl;
    int i=top;
    while (i>=0){
        cout<< "        |      "<<values[i]<<"      |"<<endl;
        i--;
    }
    cout<<"        -------------"<<endl;
}
```

Main File:

```
#include <iostream>
#include "Stack.h"
using namespace std;


int main()
{
    //making obj of stack class
    Stack s;
    s.push(1);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(2);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(3);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(4);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    cout<<"Calling pop function"<<endl;
    s.pop();
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    cout<<"Calling pop function"<<endl;
    s.pop();
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;


}
```

DIRECTORY:



main.cpp [Stack] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+
Management
Projects Files
Workspace
Stack
Sources
src
main.cpp
Headers
include
main.cpp Stack.h Stack.cpp
Logs & others
Code::Blocks Search results Cccc Build log
C:\Users\mary3\New folder (2)\Stack\main.cpp
"C:\Users\mary3\New folder (
4:29 pm
17/10/2024

```
#include <iostream>
#include "Stack.h"
using namespace std;



int main()
{
    //making obj of stack class
    Stack s(10);
    s.push(1);
    s.print();
    cout<<"Top element is: "<<s.Top()<<
    s.push(2);
    s.print();
    cout<<"Top element is: "<<s.Top()<<
    s.push(3);
    s.print();
    cout<<"Top element is: "<<s.Top()<<
    s.push(4);
    s.print();
    cout<<"Top element is: "<<s.Top()<<
    cout<<"Calling pop function"<<endl;
    s.pop();
    s.print();
    cout<<"Top element is: "<<s.Top()<<
    cout<<"Calling pop function"<<endl;
```

```
............................
Top -->
           |     2     |
           |     1     |
           -------------
Top element is: 2
............................
Top -->
           |     3     |
           |     2     |
           |     1     |
           -------------
Top element is: 3
............................
Top -->
           |     4     |
           |     3     |
           |     2     |
           |     1     |
           -------------
Top element is: 4
............................
Calling pop function
Top -->
           |     3     |
           |     2     |
           |     1     |
           -------------
Top element is: 3
............................
```

IMPLEMENTATION BY LINKED LIST:

Header File(.h File):

```
#ifndef STACK_H
#define STACK_H


struct Node{
    double data;
    Node* next;
};


class Stack{
    public:
      Stack();   //constructor
      ~Stack();  //destructor


      bool empty() const;     //function to check if stack is empty
      void push(const double x);    //function to add element to stack
      double pop();    //func to delete top element from stack
      double Top() const;  //returning top element
      void print() const;  // printing the stack

    private:
      Node* top;  //pointer to top element
};
#endif
```

Implementation File (.CPP File):

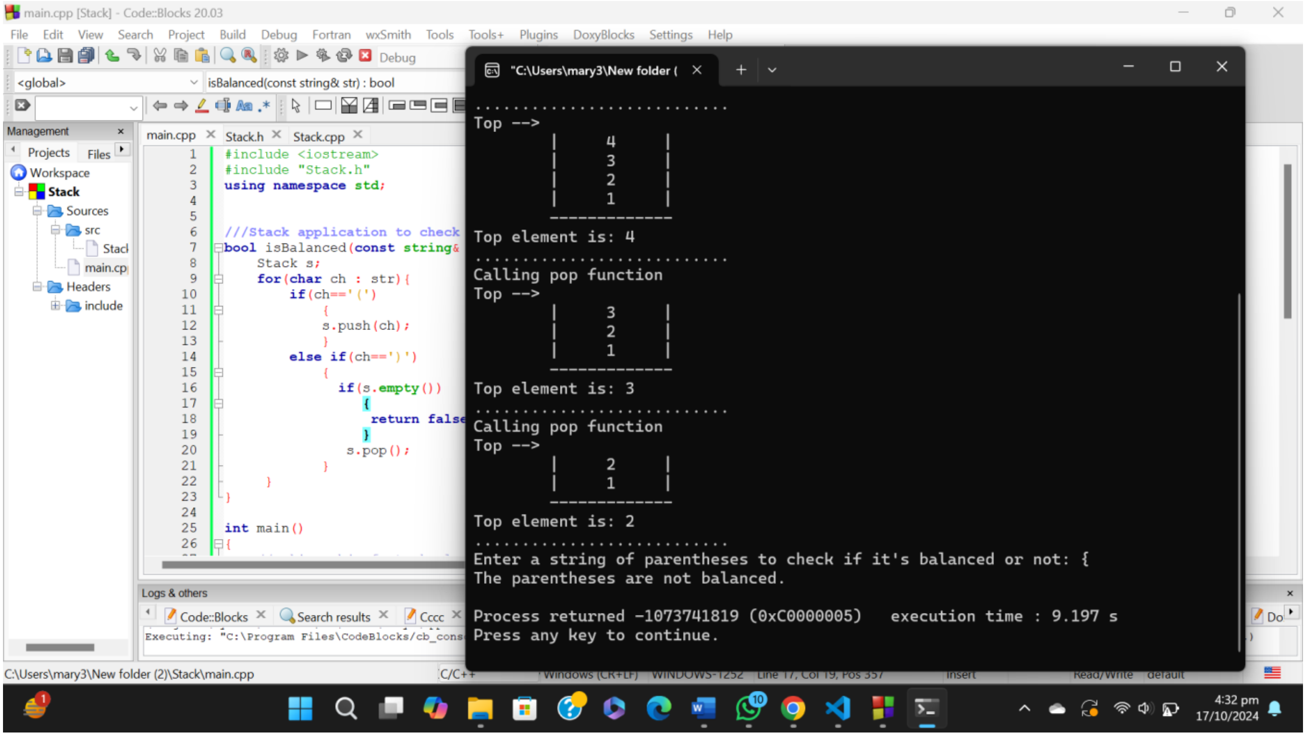
```
#include "Stack.h"
#include <iostream>
using namespace std;

Stack::Stack(){
   top=nullptr;
}

Stack::~Stack(){
    while(!empty()){
    pop();
   }
}

bool Stack::empty() const{
    if(top==nullptr){
        return true;
    }else
```

```
        return false;
}

void Stack::push(const double x){
    Node* ptr=new Node();
    ptr->data=x;
    ptr->next=top;
    top=ptr;
}

double Stack::pop(){
    if(empty()){
        cout << "Stack is empty"<< endl;
        return -1;
    }else{
       Node* ptr=top;
       top=top->next;
       delete ptr;
       return top->data;
    }
}
double Stack::Top() const{
    if(empty()){
        cout << "Stack is empty"<< endl;
        return -1;
    }else{
        return top->data;
    }
}
void Stack::print() const{
    Node* ptr=top;
    cout<<"Top --> "<<endl;

    while (ptr!=nullptr){
        cout<< "         |      "<<ptr->data<<"      |"<<endl;
        ptr=ptr->next;
    }
    cout<<"        -------------"<<endl;
}
```

Main File:

```
#include <iostream>
#include "Stack.h"
using namespace std;

///Stack application to check balance of symbols
bool isBalanced(const string& str){
    Stack s;
    for(char ch : str){
        if(ch=='(')
            {
            s.push(ch);
            }
        else if(ch==')')
            {
              if(s.empty())
                {
                  return false;
                }
               s.pop();
            }
     }
}

int main()
{
    //making obj of stack class
    Stack s;
    s.push(1);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(2);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(3);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    s.push(4);
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    cout<<"Calling pop function"<<endl;
    s.pop();
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;
    cout<<"Calling pop function"<<endl;
    s.pop();
    s.print();
    cout<<"Top element is: "<<s.Top()<<endl<<"..........................."<<endl;

    string exp;
    cout << "Enter a string of parentheses to check if it's balanced or not: ";
    cin >> exp;

    if(isBalanced(exp)) {
        cout << "The parentheses are balanced." << endl;
    } else {
        cout << "The parentheses are not balanced." << endl;
    }

    return 0;
}
```

DIRECTORY:



main.cpp [Stack] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
isBalanced(const string& str) : bool
Management
Projects Files
Workspace
Stack
Sources
src
Stacl
main.cp
Headers
include
main.cpp Stack.h Stack.cpp
Logs & others
Code::Blocks Search results Cccc
"C:\Users\mary3\New folder (
C:\Users\mary3\New folder (2)\Stack\main.cpp
C/C++
Windows (CR+LF) WINDOWS-1252 Line 17, Col 19, Pos 357
Insert
Read/Write default
4:32 pm
17/10/2024

```
#include <iostream>
#include "Stack.h"
using namespace std;


///Stack application to check
bool isBalanced(const string&
    Stack s;
    for(char ch : str){
        if(ch=='(')
        {
            s.push(ch);
        }
        else if(ch==')')
        {
            if(s.empty())
            {
                return false
            }
            s.pop();
        }
    }
}

int main()
{
```

```
Executing: "C:\Program Files\CodeBlocks/cb_cons
```

```
...........................
Top -->
         |     4     |
         |     3     |
         |     2     |
         |     1     |
         -------------
Top element is: 4
...........................
Calling pop function
Top -->
         |     3     |
         |     2     |
         |     1     |
         -------------
Top element is: 3
...........................
Calling pop function
Top -->
         |     2     |
         |     1     |
         -------------
Top element is: 2
...........................
Enter a string of parentheses to check if it's balanced or not: {
The parentheses are not balanced.

Process returned -1073741819 (0xC0000005)   execution time : 9.197 s
Press any key to continue.
```