Assignment 9

Class - SE IV
Roll NO - 21630
Batch - F4
DOS - 24/11/2020

Problem statement :

In any language program mostly syntax error occur due to unbalancing delimite such as (), {}, []. Write C++ program using stack to check whether given expression is well paranthesized or not.

Learning objectives :

1. TO implement simple C++ program.
2. TO implement stack data structure

Learning Outcomes

Will be able to implement stack data structure in C++.

S/W and H/W requirements.

1. C++ open source tool like GCC/G++.
2. Open source IDE Eclipse.
3. Windows 10 64 bit 8GB ram.

Theory.

Stack is abstract data type and linear data structure. A stack is data structure in which addition of new element or deletion of exiting element always takes place at a same end. This end is known as the the top of the stack. That means that it is possible to remove elements from a stack in a reverse order from the inseration of elements into the stack.

One other way to describing the stack is as last in first (LIFO) abstract data type and linear data structure.

Operations in Stack-:

1. Push.

Push operation refers to inserting an element in the stack. Since there' is only one position at which the new element can be inserted - Top of the stack. The new element is inserted at top of the stack.

2. Pop.

Pop operation refers to the removal of an element Again since we only have access to the element at the top of the stack, their is only one element that we can remove, we just remove the top of the stack.

We can also choose to return the value of popped element back but its the choice of programmer.

Page :
Date : / /

* Pseudocode -:

```
ADT structure for class Stack
    char data[Max] //Stores value
    int top //points to access top element of stack.

    initialize  // initializes top to -1
    push        // Inserts element in stack
    pop         // Delete top element From stack
    full        // check stack is ful or not.
    Empty       // check stack is empty or not.
```

* Void initialize()

```
    set top=-1
END
```

* void push (element)

```
    If (stack is not full)
        set top= top+1
        data[top]= element
    END If
END
```

* Void pop()

```
    If ( stack is not empty)
        ch=data[top];
        top = top-1
        return ch
    END IF
END
```

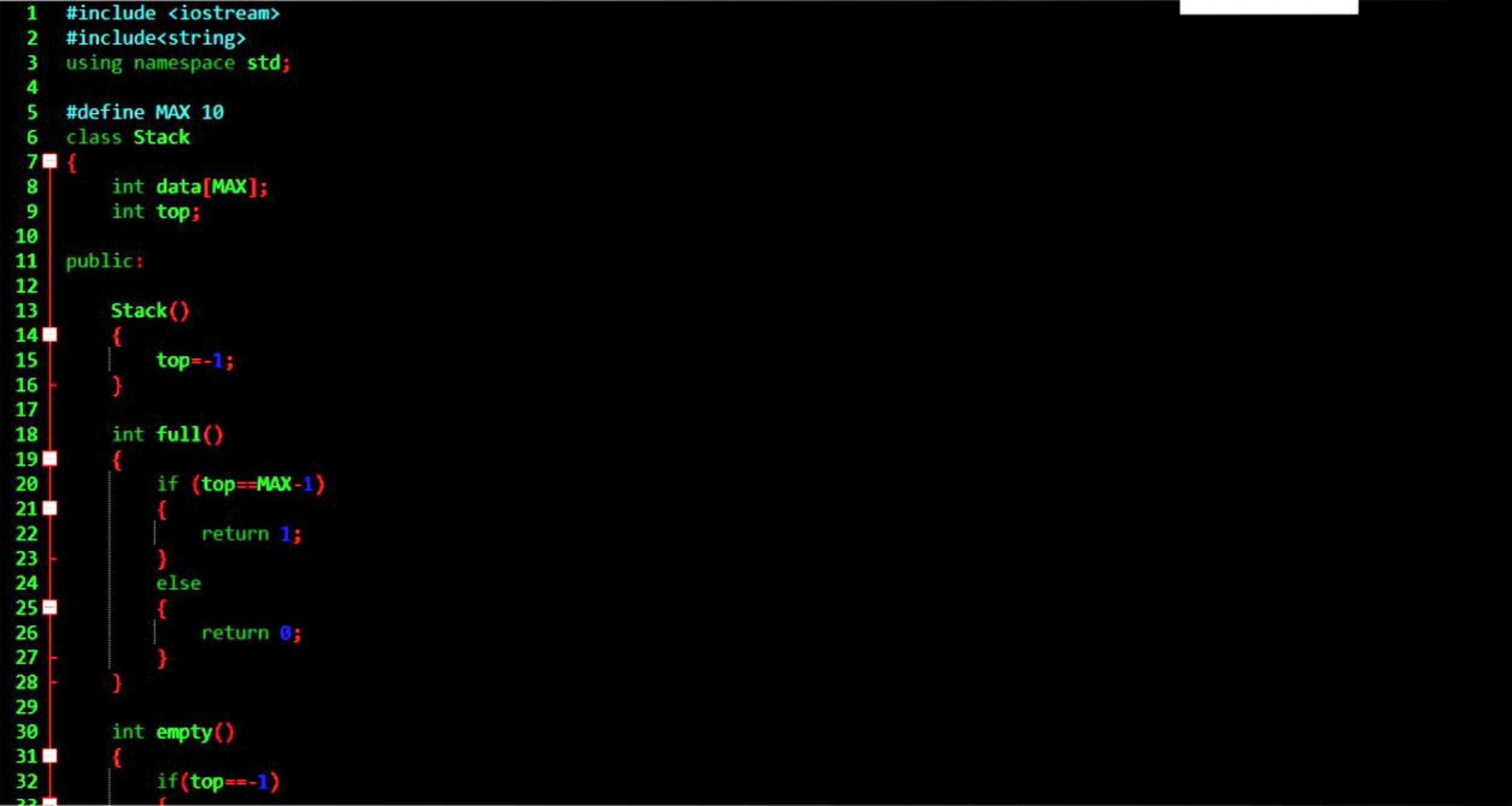
```
int full()
    if (top == max-1)
        return True
    else
        return false
END

int empty()
    if (top == -1)
        return True
    else
        return false
END

int main()
    Declare stack object
    Input ~~s~~ expression from user.
    for i from 1 to exp.length()
        ch = exp[i]
        if (ch is '(' or '{' or '[')
            s.push(ch)
        else.if ( ch is ')' or '}' or ']' and
                          stack is empty)
            break;
        else if ( ch is ')' or '}' or ']' and
                          stack is not empty)
            sch = pop()
        if (sch is '(' and ch is not ')')
            break;
            else if(sch is '}' and ch is not '{')
                break
            else if (sch is '[' and ch is not ']')
                break
            END if
      END For
      if ( Loop does not break & stack is empty)
            display Yes.
      else
            display NO.
END.
```

Test Cases -:

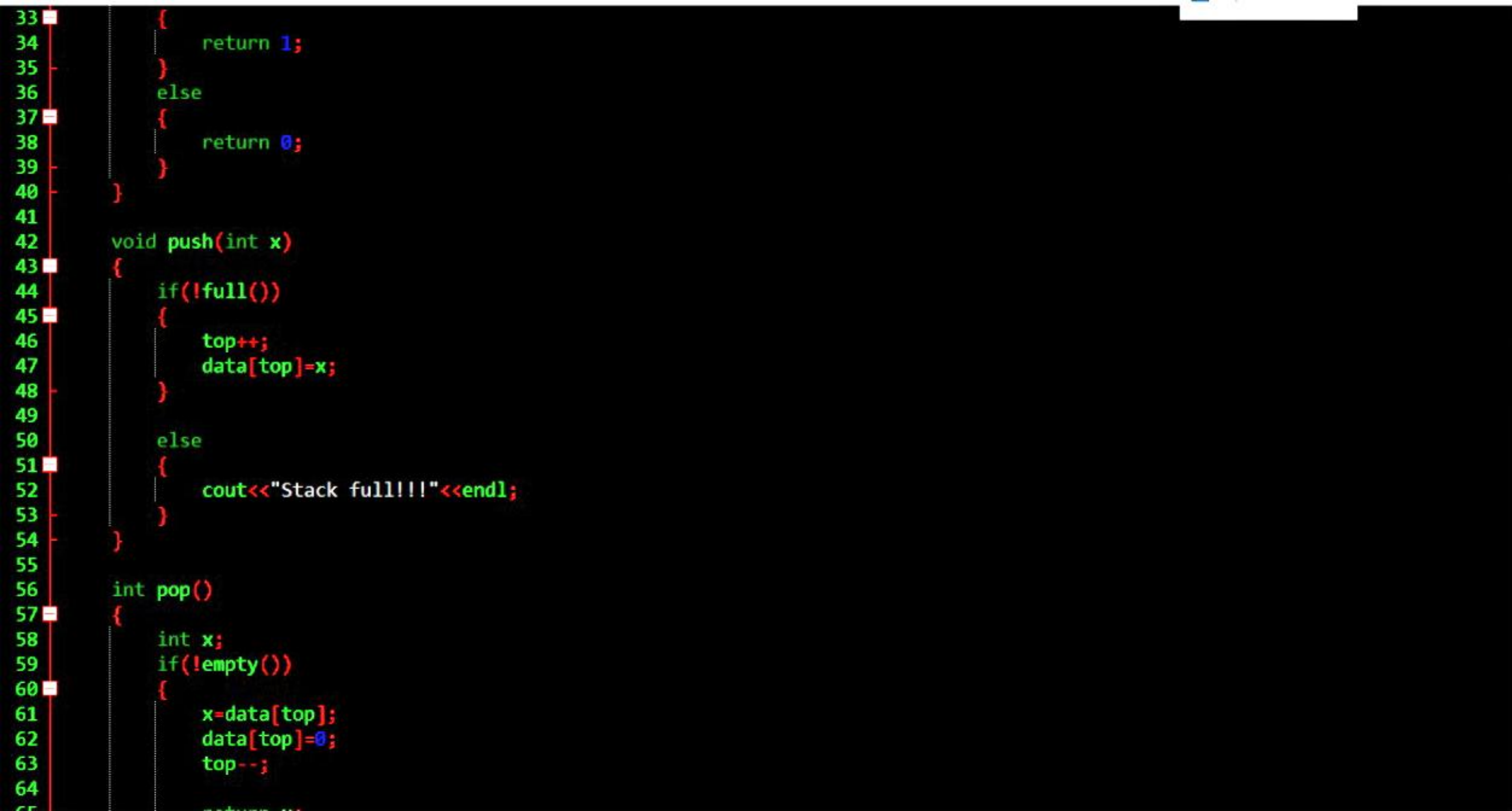
NO.	Description	Input	Expected out	Actual output	Result
1.	Enter expression	(a+b)*c	YES	YES	Pass
2.	Enter Expression	[(a+b])	NO	NO	Pass

Complexity.

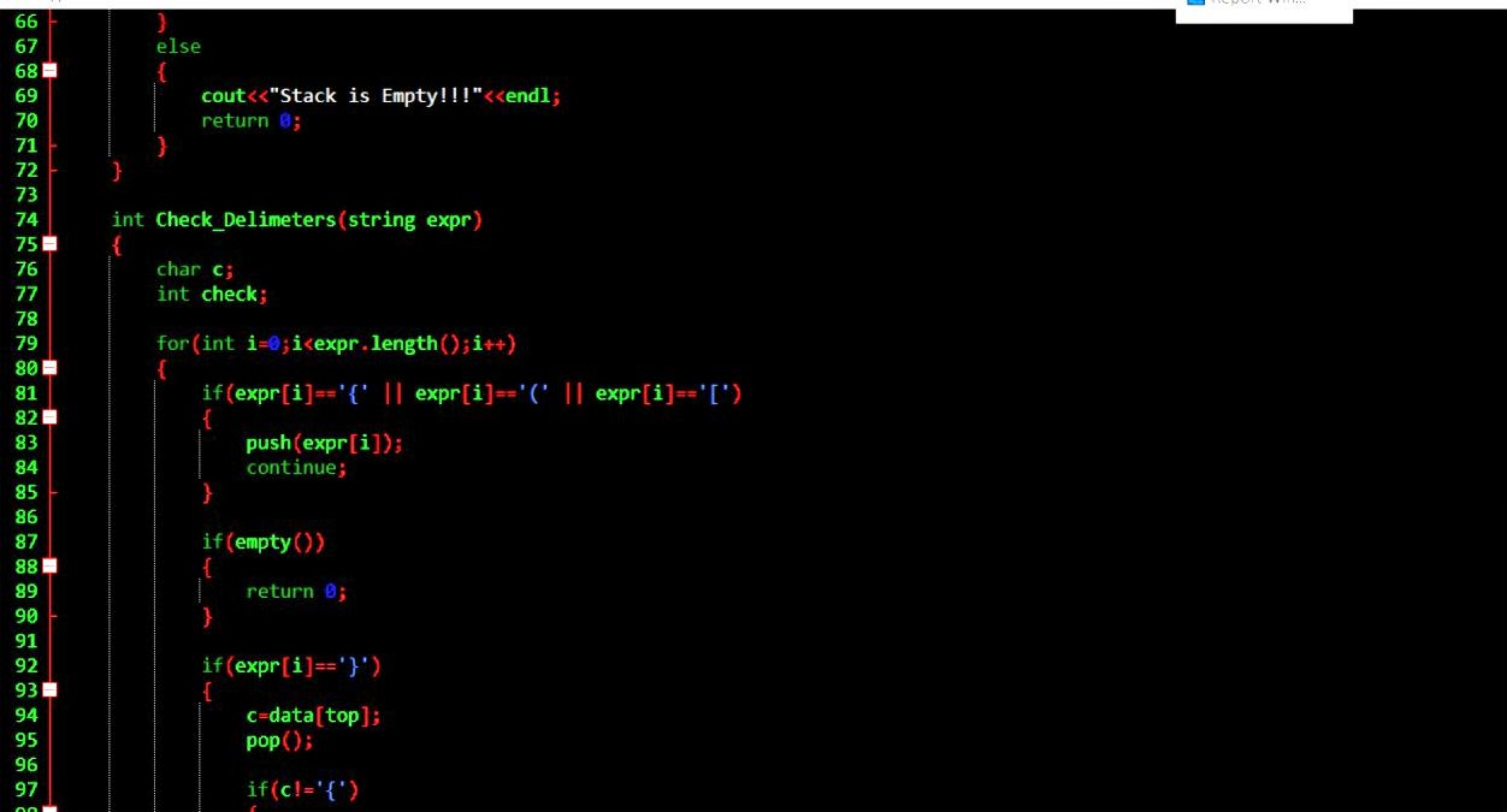
Function	Time complex	space complexity
Initialize()	O(1)	O(1)
Push(), POP()	O(1), O(1)	O(1), O(1)
Main()	O(1)	O(1)
Full()	O(1)	O(1)



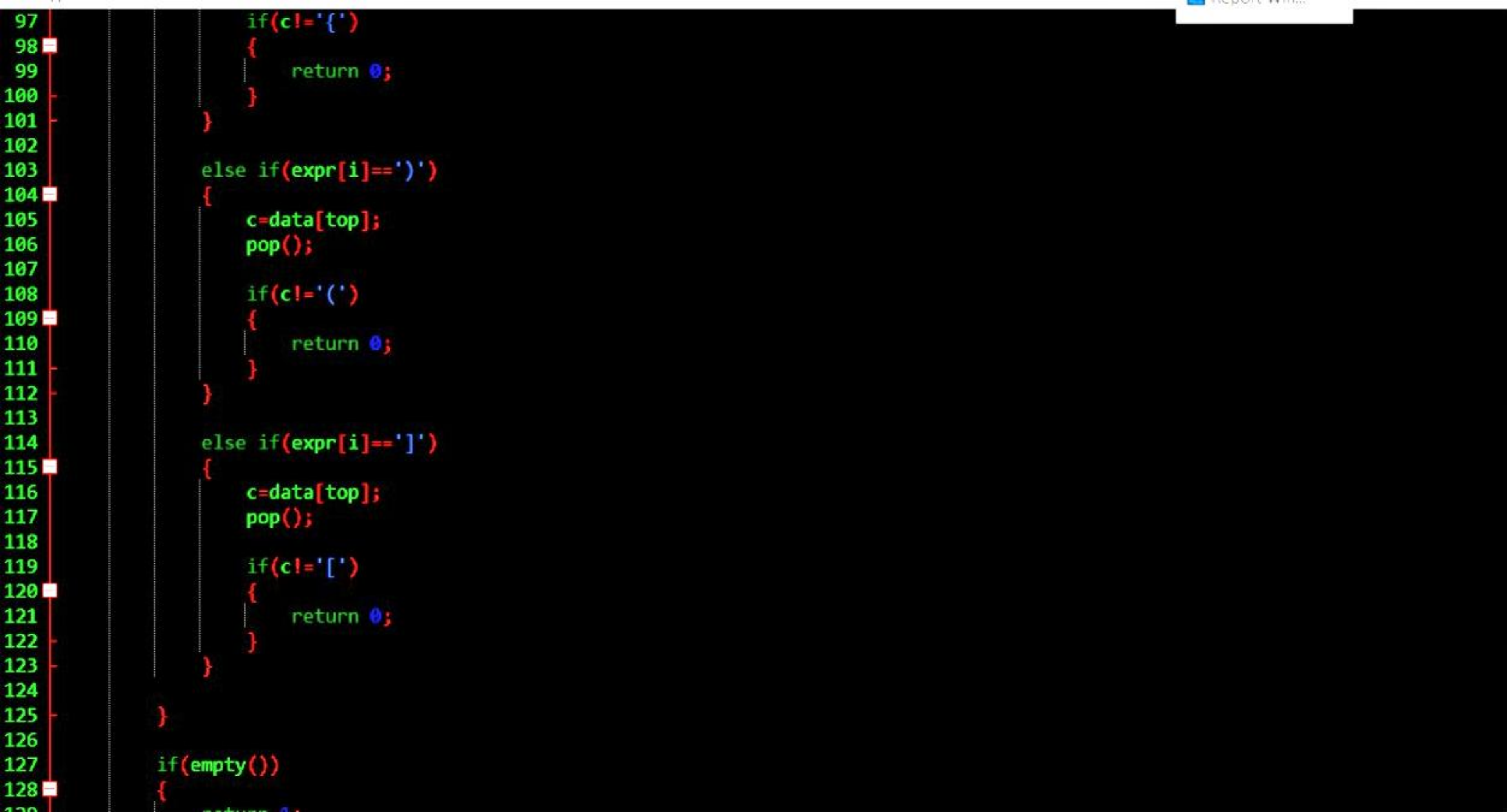
```cpp
#include <iostream>
#include<string>
using namespace std;

#define MAX 10
class Stack
{
    int data[MAX];
    int top;

public:

    Stack()
    {
        top=-1;
    }

    int full()
    {
        if (top==MAX-1)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }

    int empty()
    {
        if(top==-1)
        {
```



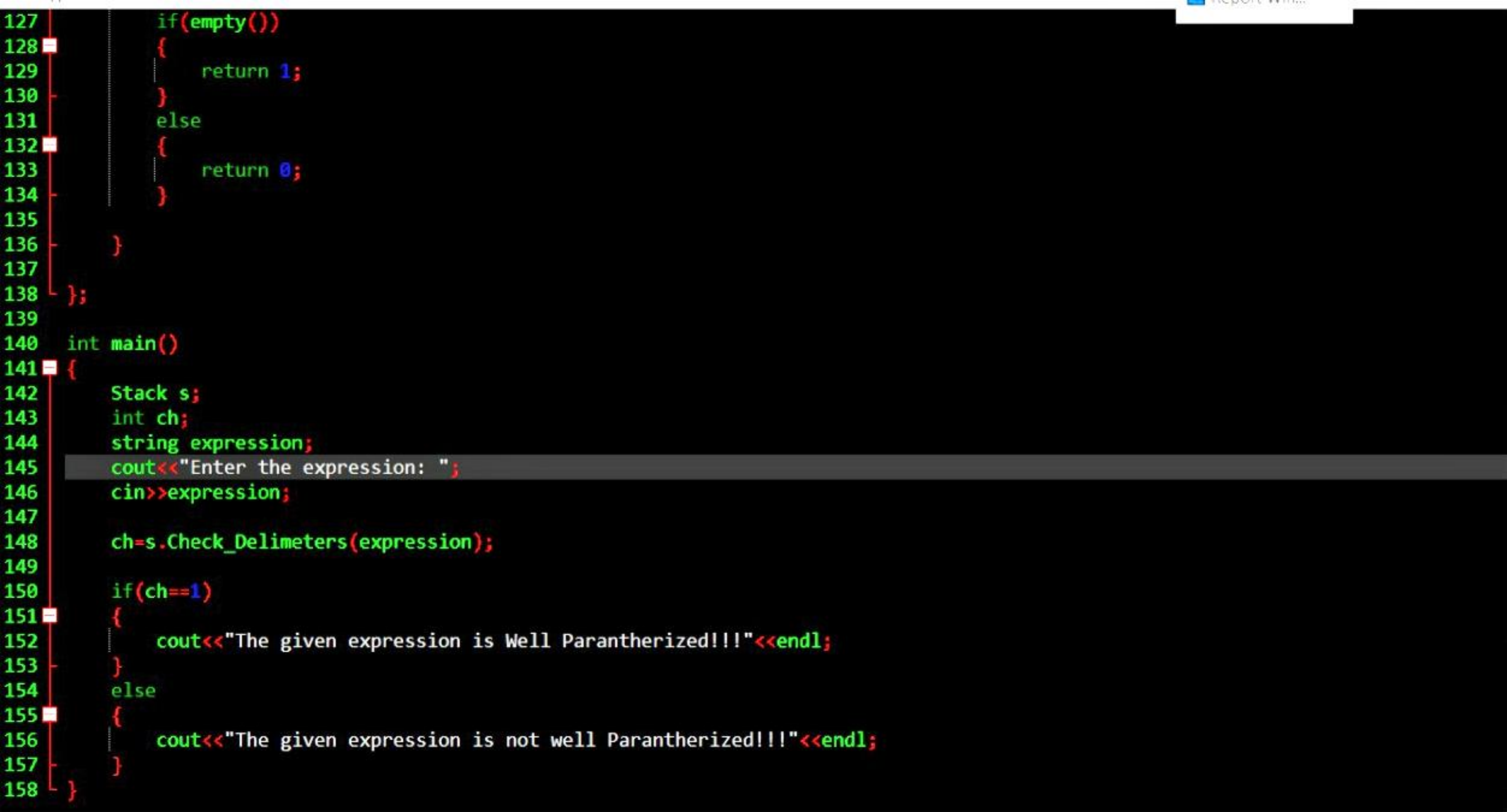
```
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }

    void push(int x)
    {
        if(!full())
        {
            top++;
            data[top]=x;
        }

        else
        {
            cout<<"Stack full!!!"<<endl;
        }
    }

    int pop()
    {
        int x;
        if(!empty())
        {
            x=data[top];
            data[top]=0;
            top--;

            return x;
```



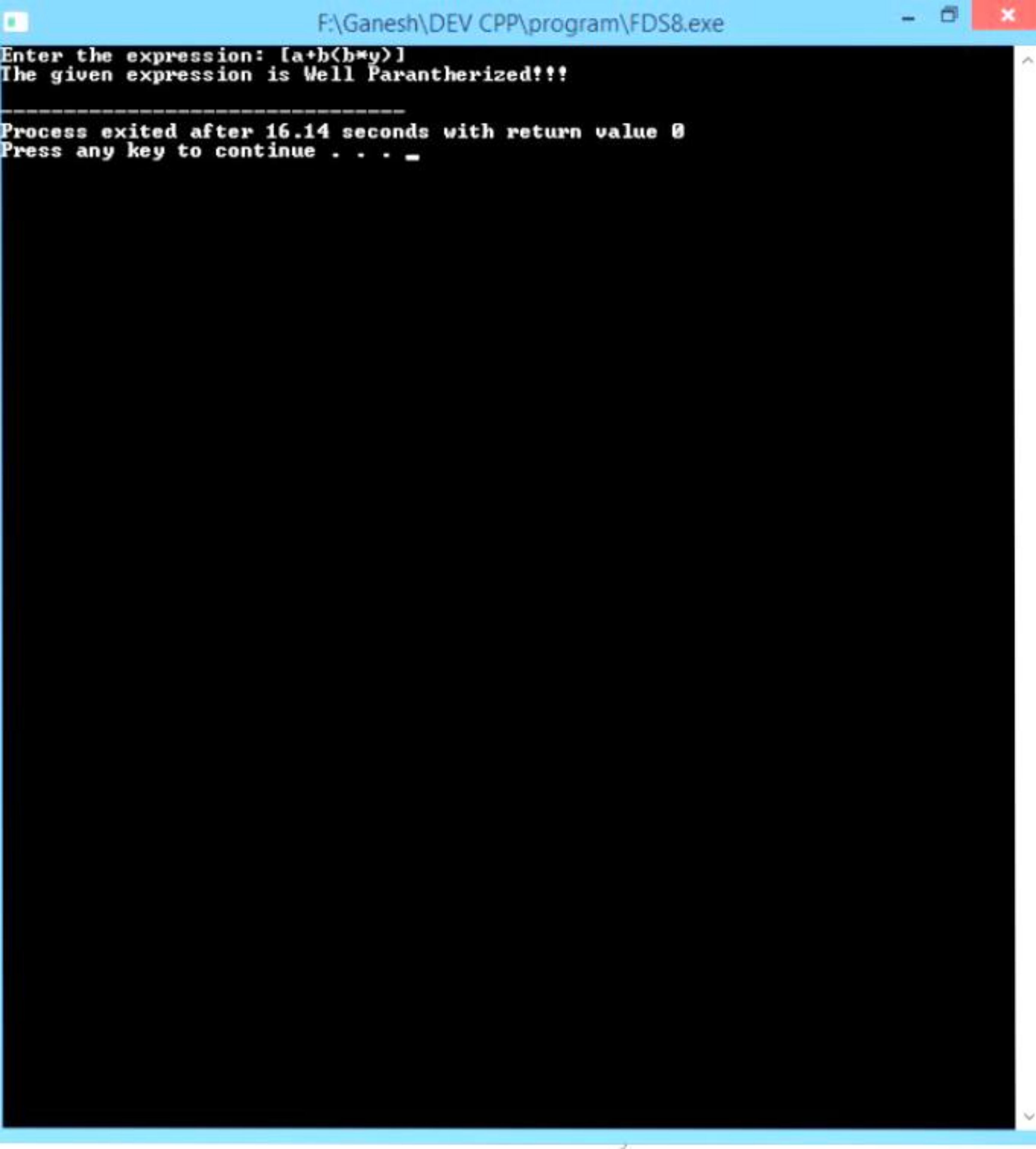
```
    }
    else
    {
        cout<<"Stack is Empty!!!"<<endl;
        return 0;
    }
}

int Check_Delimeters(string expr)
{
    char c;
    int check;

    for(int i=0;i<expr.length();i++)
    {
        if(expr[i]=='{' || expr[i]=='(' || expr[i]=='[')
        {
            push(expr[i]);
            continue;
        }

        if(empty())
        {
            return 0;
        }

        if(expr[i]=='}')
        {
            c=data[top];
            pop();

            if(c!='{')
            {
```



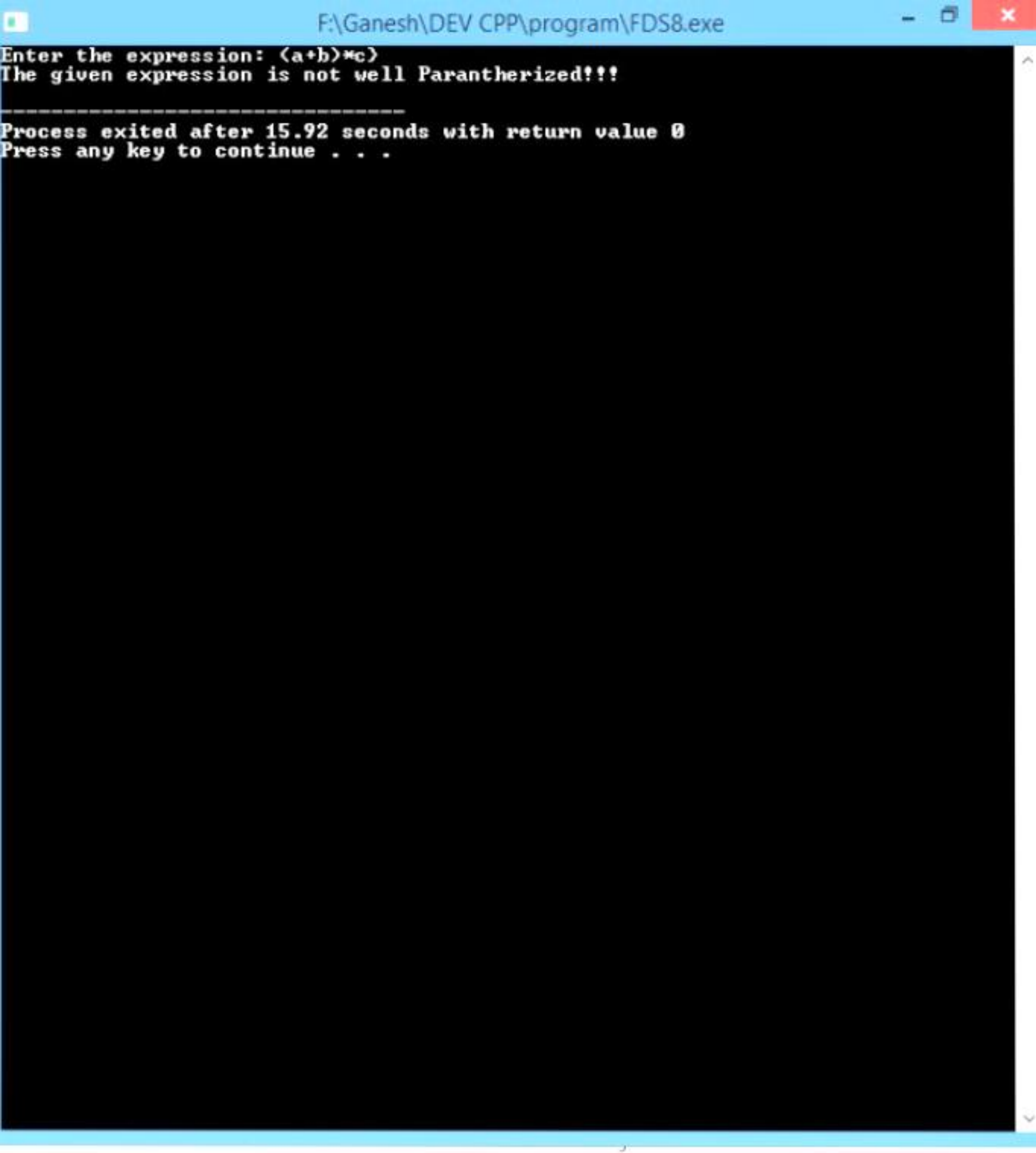
```
                if(c!='{')
                {
                    return 0;
                }
            }

            else if(expr[i]==')')
            {
                c=data[top];
                pop();

                if(c!='(')
                {
                    return 0;
                }
            }

            else if(expr[i]==']')
            {
                c=data[top];
                pop();

                if(c!='[')
                {
                    return 0;
                }
            }

        }

        if(empty())
        {
            return 0;
```



```
        if(empty())
        {
            return 1;
        }
        else
        {
            return 0;
        }

    }

};

int main()
{
    Stack s;
    int ch;
    string expression;
    cout<<"Enter the expression: ";
    cin>>expression;

    ch=s.Check_Delimeters(expression);

    if(ch==1)
    {
        cout<<"The given expression is Well Parantherized!!!"<<endl;
    }
    else
    {
        cout<<"The given expression is not well Parantherized!!!"<<endl;
    }
}
```



```
Enter the expression: [a+b(b*y)]
The given expression is Well Parantherized!!!

--------------------------------
Process exited after 16.14 seconds with return value 0
Press any key to continue . . . _
```



```
Enter the expression: (a+b)*c)
The given expression is not well Parantherized!!!

--------------------------------
Process exited after 15.92 seconds with return value 0
Press any key to continue . . .
```