Group D-26:--- Kaustubh Shrikant Kabra SE COMP-1 20

Program:---

#include <iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

#define MAX 50 /\* Size of Stack \*/

class Stack

{

char s[MAX];

int top;

public:

Stack()

{

top=-1;

}

void push(char ch);

char pop();

bool isEmpty();

bool isFull();

bool checkParenthesis(char expr[]);

};

bool Stack::isEmpty()

{

if(top==-1)

return 1;

else

return 0;

}

bool Stack::isFull()

{

if(top==MAX-1)

return 1;

else

return 0;

}

void Stack::push(char ch)

{

if(!isFull())

{

top++;

s[top]=ch;

}

}

char Stack::pop()

{

if(!isEmpty())

{

char ch=s[top];

top--;

return ch;

}

else

return '\0';

}

bool Stack::checkParenthesis(char expr[])

{

char x;

// Traversing the Expression

for (int i=0; expr[i]!='\0'; i++)

{

if (expr[i]=='('||expr[i]=='['||expr[i]=='{')

{

// Push the element in the stack

push(expr[i]);

continue;

}

// IF current current character is not opening

// bracket, then it must be closing. So stack

// cannot be empty at this point.

if (isEmpty())

return false;

switch (expr[i])

{

case ')':

// Store the top element in a

x = pop();

if (x=='{' || x=='[')

return false;

break;

case '}':

// Store the top element in b

x = pop();

if (x=='(' || x=='[')

return false;

break;

case ']':

// Store the top element in c

x = pop();

if (x =='(' || x == '{')

return false;

break;

}

}

// Check Empty Stack

return (isEmpty());

}

// Driver program to test above function

int main()

{

char expr[50];

int i=0,k=0;

Stack st;

cout<<"\nEnter Expression: ";

cin>>expr;

if (st.checkParenthesis(expr))

cout << "Balanced";

else

cout << "Not Balanced";

return 0;

}

**Output:-**

Enter Expression: (){]

Not Balanced

Enter Expression: (){}

Balanced

Enter Expression: [{(a+b+c)\*(a+b-c)}-{(a-b-c)\*(a+c-b)}]

Balanced