#include<iostream>

using namespace std;

const int MAX=20;

class Stack

{

    char str[MAX];

    int top;

    public:

        Stack()

        {

            top=-1;

        }

        void push(char ch);

        char pop();

    //  char getTop();

        bool isEmpty();

        bool isFull();

        void display();

        void checkParenthesis();

};

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 :: display()

{

    if(isEmpty()==1)

        cout<<"\nStack is empty";

    else

    {

        for(int i=0;i<=top;i++)

        {

            cout<<" "<<str[i];

        }

    }

}

void Stack::push(char ch)

{

    if(!isFull())

    {

        top++;

        str[top]=ch;

    }

}

char Stack::pop()

{

    if(!isEmpty())

    {

        char ch=str[top];

        top--;

        return ch;

    }

    else

    {

        return '\0';

    }

}

void Stack::checkParenthesis()

{

    cout<<"\nEnter # as a deliminator after expression(At the end)\n";

    cout<<"\nEnter Expression: ";

    cin.getline(str,MAX,'#');

    char ch;

    bool flag=0;

    for(int i=0;str[i]!='\0';i++)

    {

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

            push(str[i]);

        if(str[i]==')'||str[i]==']'||str[i]=='}')

        {

            ch=pop();

            if((str[i]==')'&& ch!='(') ||(str[i]==']'&& ch!='[')||(str[i]=='}'&& ch!='{'))

            {

                cout<<"\nNot parenthesized At "<<i<<" = "<<str[i];

                flag=1;

                break;

            }

        }

    }

    if(isEmpty()==1 && flag==0)

        cout<<"\nExpression is Well Parenthesized.";

    else

        cout<<"\nExpression is not Well Parenthesized.";

}

int main()

{

    int choice;

    do

    {

        Stack s;

        s.checkParenthesis();

        cout<<"\nDO you want to continue?{1/0)";

        cin>>choice;

    }while(choice!=0);

    return 0;

}

Enter # as a deliminator after expression(At the end)

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

1

Expression is Well Parenthesized.

DO you want to continue?{1/0) 0

Enter # as a deliminator after expression(At the end)

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

{

1

Not parenthesized At 14 = {

DO you want to continue?{1/0) 0