**/\*WAP to check parenthesis of algebraic expression using stack\*/**

**#include <iostream>**

**#include <cstring>**

**#define MAXSIZE 30**

**using namespace std;**

**char expression[MAXSIZE],stack[MAXSIZE];**

**int top=-1;**

**bool isempty()**

**{**

**if(top == -1)**

**return true;**

**else**

**return false;**

**}**

**bool isfull()**

**{**

**if(top == MAXSIZE)**

**return true;**

**else**

**return false;**

**}**

**void pop()**

**{**

**char check;**

**int cl\_sb=0,cr\_sb=0,cl\_cb=0,cr\_cb=0,cl\_bb=0,cr\_bb=0;**

**/\*cl\_sb count left of small barcket,cr\_sb count right of small barcket\*/**

**/\*cl\_cb count left of curly barcket,cr\_cb count right of curly barcket\*/**

**/\*cl\_bb count left of big barcket,cr\_bb count right of big barcket\*/**

**while (!isempty())**

**{**

**check=stack[top--];**

**if (check=='(')**

**cl\_sb++;**

**if (check==')')**

**cr\_sb++;**

**if (check=='{')**

**cl\_cb++;**

**if (check=='}')**

**cr\_cb++;**

**if (check=='[')**

**cl\_bb++;**

**if (check==']')**

**cr\_bb++;**

**}**

**if ((cl\_sb=cr\_sb) && (cl\_cb==cr\_cb) && (cl\_bb==cr\_bb))**

**cout<<"Expression is balanced.\n";**

**else cout<<"Expression is not balanced.\n";**

**}**

**void push()**

**{**

**cout<<"Enter expression: ";**

**cin>>expression;**

**for(int i=0; i<strlen(expression); i++)**

**{**

**if (!isfull())**

**{**

**stack[++top]=expression[i];**

**}**

**else**

**cout<<"Expression overflow!!!";**

**}**

**}**

**int main()**

**{**

**while (1)**

**{**

**push();**

**cout<<"\n";**

**pop();**

**cout<<"\n";**

**}**

**return 0;**

**}**

**/\*WAP to check parenthesis of algebraic expression using stack\*///or**

**#include<iostream>**

**#define max 10**

**using namespace std;**

**template<class T>**

**class Stack**

**{**

**T data[max];**

**int top;**

**public:**

**Stack():top(-1) {}**

**void push(T value)**

**{**

**if(top==max-1)**

**{**

**cout<<"overflow"<<endl;**

**}**

**else**

**data[++top]=value;**

**}**

**T pop()**

**{**

**if(top==-1)**

**{**

**//cout<<"underflow"<<endl;**

**return '0';**

**}**

**else**

**{**

**return data[top--];**

**}**

**}**

**void peek()**

**{**

**if(top==-1)**

**{**

**cout<<"underflow"<<endl;**

**}**

**else**

**{**

**cout<<data[top]<<" is in top"<<endl;**

**}**

**}**

**void display()**

**{**

**cout<<"------------------XX---------------"<<endl;**

**for(int i=top; i>-1; i--)**

**{**

**cout<<data[i]<<endl;**

**}**

**cout<<"------------------XX---------------"<<endl;**

**if(top==-1)**

**{**

**cout<<"expression is correct"<<endl;**

**}**

**else**

**{**

**cout<<"error in expression: "<<top+1<<" \")\" is missing"<<endl;**

**}**

**}**

**};**

**int main()**

**{**

**Stack<char> arr;**

**char test;**

**string exp;**

**cout<<"enter an expression: ";**

**getline(cin,exp);**

**int i=0;**

**while(exp[i]!='\0')**

**{**

**if(exp[i]=='(')**

**{**

**arr.push(exp[i]);**

**}**

**else if(exp[i]==')')**

**{**

**test=arr.pop();**

**if(test=='0')**

**{**

**cout<<"No \"(\" to pop !!empty Stack"<<endl;**

**}**

**}**

**i++;**

**}**

**arr.display();**

**return 0;**

**}**