/\*

as (),{},[]. In a given expression while (m<(n[8]+0)), the first opening parenthesis matched with the last closing parenthesis, but this is done after the second opening parenthesis is matched with the next to last closing parenthesis, this, in turn, is done after the opening square bracket is matched with the closing bracket. Here, identify the data structure needed to implement delimiter matching program to display the given statement is balanced/ well- parenthesis or not. Write C++ program using stack to check whether given expression is well parenthesized or not

\*/

#include<iostream>

#include<string.h>

#define MAX 20

using namespace std;

typedef struct stack

{

char data[MAX];

int top;

}stack;

void init(stack \*stck)

{

int i;

for(i=0;i<MAX;i++)

stck->data[i]='\0';

stck->top=-1;

}

int isempty(stack stck)

{

return stck.top==-1?1:0;

}

int isfull(stack stck)

{

return stck.top==MAX-1?1:0;

}

void push(stack \*stck,char data)

{

stck->top+=1;

stck->data[stck->top]=data;

}

char pop(stack \*stck)

{

char data;

data=stck->data[stck->top];

stck->data[stck->top]='\0';

stck->top-=1;

return data;

}

int main()

{

stack stck;

int i,found=0;

char data[MAX],ch;

init(&stck);

cout<<"\nEnter the expression";

cin>>data;

for(i=0;i<strlen(data);i++)

{

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

push(&stck,data[i]);

else

{

switch(data[i])

{

case ')':

if(isempty(stck))

{

cout<<"\nInvalid expression";

}

ch=pop(&stck);

if((ch=='[')||(ch=='}'))

{

cout<<"\nInvalid Expression";

found=1;

}

break;

case ']':

if(isempty(stck))

{

cout<<"\nInvalid Expression";

found=1;

}

ch=pop(&stck);

if((ch=='(')||(ch=='{'))

{

cout<<"\nInva;id Expression";

found=1;

}

break;

case '}':

if(isempty(stck))

{

cout<<"\nInvalid Expression";

found=1;

}

ch=pop(&stck);

if((ch=='{')||(ch=='('))

{

cout<<"\nInvalid Expression";

found=1;

}

break;

}

}

}

if(found==0)

{

if(isempty(stck))

cout<<"Valid Expression";

else

cout<<"\nInvalid Expression";

}

return 0;

}

/\*

OUTPUT :

Enter the expression (a+b)

Valid Expression

\*/