/\*\*

In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[].

Write C++ program using stack to check whether given expression is well parenthesized or not

\*\*/

#include<stdio.h>

#include<stdlib.h>

#include<iostream>

using namespace std;

#define bool int

/\* structure of a stack node \*/

struct sNode

{

char data;

struct sNode \*next;

};

/\* Function to push an item to stack\*/

void push(struct sNode\*\* top\_ref, int new\_data);

/\* Function to pop an item from stack\*/

int pop(struct sNode\*\* top\_ref);

/\* Returns 1 if character1 and character2 are matching left

and right Parenthesis \*/

bool isMatchingPair(char character1, char character2)

{

if (character1 == '(' && character2 == ')')

return 1;

else if (character1 == '{' && character2 == '}')

return 1;

else if (character1 == '[' && character2 == ']')

return 1;

else

return 0;

}

/\*Return 1 if expression has balanced Parenthesis \*/

bool areParenthesisBalanced(char exp[])

{

int i = 0;

/\* Declare an empty character stack \*/

struct sNode \*stack = NULL;

/\* Traverse the given expression to check matching parenthesis \*/

while (exp[i])

{

/\*If the exp[i] is a starting parenthesis then push it\*/

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

push(&stack, exp[i]);

/\* If exp[i] is a ending parenthesis then pop from stack and

check if the popped parenthesis is a matching pair\*/

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

{

/\*If we see an ending parenthesis without a pair then return false\*/

if (stack == NULL)

return 0;

/\* Pop the top element from stack, if it is not a pair

parenthesis of character then there is a mismatch.

This happens for expressions like {(}) \*/

else if ( !isMatchingPair(pop(&stack), exp[i]) )

return 0;

}

i++;

}

/\* If there is something left in expression then there is a starting

parenthesis without a closing parenthesis \*/

if (stack == NULL)

return 1; /\*balanced\*/

else

return 0; /\*not balanced\*/

}

/\* UTILITY FUNCTIONS \*/

/\*driver program to test above functions\*/

int main()

{

char exp[100] ;

cout<<"enter:";

cin>>exp;

if (areParenthesisBalanced(exp))

printf("\n Balanced ");

else

printf("\n Not Balanced ");

return 0;

}

/\* Function to push an item to stack\*/

void push(struct sNode\*\* top\_ref, int new\_data)

{

/\* allocate node \*/

struct sNode\* new\_node =

(struct sNode\*) malloc(sizeof(struct sNode));

if (new\_node == NULL)

{

printf("Stack overflow \n");

getchar();

exit(0);

}

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*top\_ref);

/\* move the head to point to the new node \*/

(\*top\_ref) = new\_node;

}

/\* Function to pop an item from stack\*/

int pop(struct sNode\*\* top\_ref)

{

char res;

struct sNode \*top;

/\*If stack is empty then error \*/

if (\*top\_ref == NULL)

{

printf("Stack overflow \n");

getchar();

exit(0);

}

else

{

top = \*top\_ref;

res = top->data;

\*top\_ref = top->next;

free(top);

return res;

}

}

----------------------------------------------------------------------------------------------------------------------------------

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

enter:(a+b(b\*c-f))

Balanced

enter:(a+b(b\*c-f)

Not Balanced

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/