```c
#include <stdio.h>
#include <stdlib.h>

typedef char StackElementType;

typedef struct StackNode *StackPointer;
typedef struct StackNode
{
    StackElementType Data;
    StackPointer Next;
} StackNode;

typedef enum {
    FALSE, TRUE
} boolean;

void CreateStack(StackPointer *Stack);
boolean EmptyStack(StackPointer Stack);
void Push(StackPointer *Stack, StackElementType Item);
void Pop(StackPointer *Stack, StackElementType *Item);

//?????? ?? ???? ?? hints ??? ??????? ??? ??????.
int main()
{
    StackPointer AStack;
    int i;
    char str[40];
    boolean found;

    CreateStack(&AStack);

    printf("EISAGETE MIA PARASTASI:");
    gets(str);

    found=TRUE;

    for(i=0; str[i] != '\0'; i++)
    {
        if(str[i] == '(' || str[i] == '{' || str[i] == '[')
        {
            Push(&AStack,str[i]);
        }
        else if(str[i] == ')' || str[i] == '}' || str[i] == ']')
        {
            if(EmptyStack(AStack))
            {
                found=FALSE;
                break;
            }
            else
            {
                Pop(&AStack,&str[i]);
                if(str[i]=='}' && AStack->Data != '{')
                {
                    found=FALSE;
                    break;
                }
                else if(str[i]==')' && AStack->Data != '(')
                {
                    found=FALSE;
                    break;
                }
                else if (str[i]==']' && AStack->Data != '[')
                {
                    found=FALSE;
                    break;
```

```
                }
            }
        }
    }


    if(found==FALSE || !EmptyStack(AStack))
        printf("WRONG\n");
    else
        printf("CORRECT\n");

    return 0;
}

void CreateStack(StackPointer *Stack)
{
    *Stack = NULL;
}

boolean EmptyStack(StackPointer Stack)
{
    return (Stack==NULL);
}

void Push(StackPointer *Stack, StackElementType Item)
{
    StackPointer TempPtr;

    TempPtr= (StackPointer)malloc(sizeof(struct StackNode));
    TempPtr->Data = Item;
    TempPtr->Next = *Stack;
    *Stack = TempPtr;
}

void Pop(StackPointer *Stack, StackElementType *Item)
{
    StackPointer TempPtr;

    if (EmptyStack(*Stack)) {
        printf("EMPTY Stack\n");
    }
   else
   {
        TempPtr = *Stack;
        *Item=TempPtr->Data;
        *Stack = TempPtr->Next;
        free(TempPtr);
    }
}
```