

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef char StackElementType;
5
6  typedef struct StackNode *StackPointer;
7  typedef struct StackNode
8  {
9      StackElementType Data;
10     StackPointer Next;
11 } StackNode;
12
13 typedef enum {
14     FALSE, TRUE
15 } boolean;
16
17 void CreateStack(StackPointer *Stack);
18 boolean EmptyStack(StackPointer Stack);
19 void Push(StackPointer *Stack, StackElementType Item);
20 void Pop(StackPointer *Stack, StackElementType *Item);
21
22 //?????? ?? ???? ?? hints ??? ??????? ??? ??????.
23 int main()
24 {
25     StackPointer AStack;
26     int i;
27     char str[40];
28     boolean found;
29
30     CreateStack(&AStack);
31
32     printf("EISAGETE MIA PARASTASI:");
33     gets(str);
34
35     found=TRUE;
36
37     for(i=0; str[i] != '\0'; i++)
38     {
39         if(str[i] == '(' || str[i] == '{' || str[i] == '[')
40         {
41             Push(&AStack, str[i]);
42         }
43         else if(str[i] == ')' || str[i] == '}' || str[i] == ']')
44         {
45             if(EmptyStack(AStack))
46             {
47                 found=FALSE;
48                 break;
49             }
50             else
51             {
52                 Pop(&AStack, &str[i]);
53                 if(str[i]=='}' && AStack->Data != '{')
54                 {
55                     found=FALSE;
56                     break;
57                 }
58                 else if(str[i]==')' && AStack->Data != '(')
59                 {
60                     found=FALSE;
61                     break;
62                 }
63                 else if (str[i]==']' && AStack->Data != '[')
64                 {
65                     found=FALSE;
66                     break;

```

```

67         }
68     }
69 }
70 }
71
72
73     if(found==FALSE || !EmptyStack(AStack))
74         printf("WRONG\n");
75     else
76         printf("CORRECT\n");
77
78     return 0;
79 }
80
81 void CreateStack(StackPointer *Stack)
82 {
83     *Stack = NULL;
84 }
85
86 boolean EmptyStack(StackPointer Stack)
87 {
88     return (Stack==NULL);
89 }
90
91 void Push(StackPointer *Stack, StackElementType Item)
92 {
93     StackPointer TempPtr;
94
95     TempPtr= (StackPointer)malloc(sizeof(struct StackNode));
96     TempPtr->Data = Item;
97     TempPtr->Next = *Stack;
98     *Stack = TempPtr;
99 }
100
101 void Pop(StackPointer *Stack, StackElementType *Item)
102 {
103     StackPointer TempPtr;
104
105     if (EmptyStack(*Stack)) {
106         printf("EMPTY Stack\n");
107     }
108     else
109     {
110         TempPtr = *Stack;
111         *Item=TempPtr->Data;
112         *Stack = TempPtr->Next;
113         free(TempPtr);
114     }
115 }

```