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

struct stack
{
    int size;
    int top;
    char *arr;
};

int isEmpty(struct stack *ptr)
{
    if (ptr->top == -1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int isFull(struct stack *ptr)
{
    if (ptr->top == ptr->size - 1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

void push(struct stack* ptr, char val){
    if(isFull(ptr)){
        printf("Stack Overflow! Cannot push %d to the stack\n", va
    }
    else{
```

```
40:         ptr->top++;
41:         ptr->arr[ptr->top] = val;
42:     }
43: }
44:
45: char pop(struct stack* ptr){
46:     if(isEmpty(ptr)){
47:         printf("Stack Underflow! Cannot pop from the stack\n");
48:         return -1;
49:     }
50:     else{
51:         char val = ptr->arr[ptr->top];
52:         ptr->top--;
53:         return val;
54:     }
55: }
56:
57: char stackTop(struct stack* sp){
58:     return sp->arr[sp->top];
59: }
60:
61: int match(char a, char b){
62:     if(a=='{' && b=='}'){
63:         return 1;
64:     }
65:     if(a=='(' && b==')'){
66:         return 1;
67:     }
68:     if(a=='[' && b==']'){
69:         return 1;
70:     }
71:   return 0;
72: }
73:
74: int parenthesisMatch(char * exp){
75:     // Create and initialize the stack
76:     struct stack* sp;
77:     sp->size = 100;
78:     sp->top = -1;
```

```
79:     sp->arr = (char *)malloc(sp->size * sizeof(char));
80:     char popped_ch;
81:
82:     for (int i = 0; exp[i]!='\0'; i++)
83:     {
84:         if(exp[i]=='(' || exp[i]=='{' || exp[i]=='['){
85:             push(sp, exp[i]);
86:         }
87:         else if(exp[i]==')'|| exp[i]=='}' || exp[i]==']'){
88:             if(isEmpty(sp)){
89:                 return 0;
90:             }
91:             popped_ch = pop(sp);
92:             if(!match(popped_ch, exp[i])){
93:               return 0;
94:             }
95:         }
96:     }
97:
98:     if(isEmpty(sp)){
99:         return 1;
100:    }
101:    else{
102:        return 0;
103:    }
104:
105: }
106:
107: int main()
108: {
109:     char * exp = "[4-6]((8){(9-8)})";
110:
111:     if(parenthesisMatch(exp)){
112:         printf("The parenthesis is balanced");
113:     }
114:     else{
115:         printf("The parenthesis is not balanced");
116:     }
117:     return 0;
```

```
118: }
119:
```