```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", val
    }
    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: int parenthesisMatch(char * exp){
58:     // Create and initialize the stack
59:     struct stack* sp;
60:     sp->size = 100;
61:     sp->top = -1;
62:     sp->arr = (char *)malloc(sp->size * sizeof(char));
63:
64:
65:     for (int i = 0; exp[i]!='\0'; i++)
66:     {
67:         if(exp[i]=='('){
68:             push(sp, '(');
69:         }
70:         else if(exp[i]==')'){
71:             if(isEmpty(sp)){
72:                 return 0;
73:             }
74:             pop(sp);
75:         }
76:     }
77:
78:     if(isEmpty(sp)){
```

```
79:         return 1;
80:     }
81:     else{
82:         return 0;
83:     }
84:
85: }
86: int main()
87: {
88:     char * exp = "((8)(*--$$9))";
89:     // Check if stack is empty
90:     if(parenthesisMatch(exp)){
91:         printf("The parenthesis is matching");
92:     }
93:     else{
94:         printf("The parenthesis is not matching");
95:     }
96:     return 0;
97: }
98:
```