```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

#define MAX_SIZE 100

/* Function to check if a character is an opening parenthesis */
bool IsOpenP(char ch)
{
    return (ch == '(' || ch == '[' || ch == '{');
}

/* Function to check if two characters form a valid pair of parentheses */
bool IsValidPair(char open, char close)
{
    return (open == '(' && close == ')') ||
           (open == '[' && close == ']') ||
           (open == '{' && close == '}');
}

/* Function to validate balanced parentheses */
bool IsBalancedParentheses(const char *expression)
{
    char stack[MAX_SIZE];
    int top = -1; /* Initialize stack top */
    int i = 0;

    for (; expression[i] != '\0'; i++)
    {
        if (IsOpenP(expression[i]))
        {
            if (top < MAX_SIZE - 1)
            {
                stack[++top] = expression[i];
            }
            else
            {
                return false; /* Stack is full, cannot push more elements */
            }
        }
        else
        {
            if (top >= 0 && IsValidPair(stack[top], expression[i]))
            {
                top--; /* Pop the matching opening parenthesis */
            }
            else
            {
                return false; /* Mismatched or stack is empty */
            }
        }
    }

    return (top == -1); /* Stack should be empty for balanced parentheses */
}

int main()
{
    const char *expression = "{[()]}";
```

```
    if (IsBalancedParentheses(expression))
    {
        printf("Balanced parentheses\n");
    }
    else
    {
        printf("Unbalanced parentheses\n");
    }

    return 0;
}
```