```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100
char stack[MAX];
int top = -1;
void push(char x) {
    if (top == MAX - 1) {
        printf("Stack Overflow\n");
    } else {
        stack[++top] = x;
    }
}
char pop() {
    if (top == -1) {
        return '\0';  // Use '\0' instead of -1 for char
    } else {
        return stack[top--];
    }
}
int match(char open, char close) {
    if (open == '(' && close == ')') return 1;
    if (open == '{' && close == '}') return 1;
    if (open == '[' && close == ']') return 1;
    return 0;
}
int isBalanced(char exp[]) {
    top = -1; // reset stack for each expression
    for (int i = 0; exp[i] != '\0'; i++) {
        char ch = exp[i];
```

```
        if (ch == '(' || ch == '{' || ch == '[') {
            push(ch);
        }
        else if (ch == ')' || ch == '}' || ch == ']') {
            if (top == -1) return 0; // No opening bracket
            char popped = pop();
            if (!match(popped, ch)) return 0; // Mismatch
        }
    }
    if (top == -1) return 1;
    else return 0;
}

int main() {
    int t;
    printf("Enter number of test cases: ");
    scanf("%d", &t);

    for (int i = 0; i < t; i++) {
        char exp[MAX];
        printf("\nEnter expression %d: ", i + 1);
        scanf("%s", exp);

        if (isBalanced(exp))
            printf("The expression is well parenthesized.\n");
        else
            printf("The expression is NOT well parenthesized.\n");
    }

    return 0;
}
```

Output Clear

```
Enter number of test cases: 2

Enter expression 1: [] {}
The expression is well parenthesized.

Enter expression 2: The expression is well parenthesized.


=== Code Execution Successful ===
```