```c
// C program to check for balanced brackets.
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

#define MAX 100

// Stack structure definition
typedef struct {
    char arr[MAX];
    int top;
} Stack;

// Initialize stack
void init(Stack *s) {
    s->top = -1;
}

// Check if stack is empty
bool isEmpty(Stack *s) {
    return s->top == -1;
}

// Push element onto stack
void push(Stack *s, char c) {
    s->arr[++(s->top)] = c;
}

// Pop element from stack
char pop(Stack *s) {
    return s->arr[(s->top)--];
}

// Peek top element of stack
char peek(Stack *s) {
    return s->arr[s->top];
}

// Check if brackets are balanced
bool ispar(const char *s) {
    Stack stk;
    init(&stk);

    for (int i = 0; s[i] != '\0'; i++) {
        char ch = s[i];

        if (isEmpty(&stk)) {
            // Stack is empty, push the current bracket
            push(&stk, ch);
        } else if ((peek(&stk) == '(' && ch == ')') ||
                   (peek(&stk) == '{' && ch == '}') ||
                   (peek(&stk) == '[' && ch == ']')) {
            // Found a complete pair of brackets, pop it
            pop(&stk);
```

```
        } else {
            // Push current bracket onto stack
            push(&stk, ch);
        }
    }

    // If stack is empty, brackets are balanced
    return isEmpty(&stk);
}

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

    // Function call to check for balanced brackets
    if (ispar(s)) {
        printf("true\n");
    } else {
        printf("false\n");
    }

    return 0;
}
```