Write a program to check if a given expression is correctly parenthesized using Stacks.

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define MAX\_SIZE 100

struct Stack {

char array[MAX\_SIZE];

int top;

};

void initStack(struct Stack\* stack) {

stack->top = -1;

}

bool isEmpty(struct Stack\* stack) {

return stack->top == -1;

}

bool isFull(struct Stack\* stack) {

return stack->top == MAX\_SIZE - 1;

}

void push(struct Stack\* stack, char value) {

if (isFull(stack)) {

printf("Error: Stack is full\n");

return;

}

stack->top++;

stack->array[stack->top] = value;

}

char pop(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Error: Stack is empty\n");

return '\0';

}

char value = stack->array[stack->top];

stack->top--;

return value;

}

char peek(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Error: Stack is empty\n");

return '\0';

}

return stack->array[stack->top];

}

bool is\_balanced(char\* expression) {

int length = strlen(expression);

struct Stack stack;

initStack(&stack);

int i;

for (i = 0; i < length; i++) {

if (expression[i] == '(' || expression[i] == '[' || expression[i] == '{') {

push(&stack, expression[i]);

} else if (expression[i] == ')' || expression[i] == ']' || expression[i] == '}') {

if (isEmpty(&stack)) {

return false;

} else if (expression[i] == ')' && peek(&stack) == '(') {

pop(&stack);

} else if (expression[i] == ']' && peek(&stack) == '[') {

pop(&stack);

} else if (expression[i] == '}' && peek(&stack) == '{') {

pop(&stack);

} else {

return false;

}

}

}

return isEmpty(&stack);

}

int main() {

char expression[MAX\_SIZE];

printf("Enter an expression: ");

fgets(expression, MAX\_SIZE, stdin);

if (is\_balanced(expression)) {

printf("The expression is correctly parenthesized\n");

} else {

printf("The expression is not correctly parenthesized\n");

}

return 0;

}



