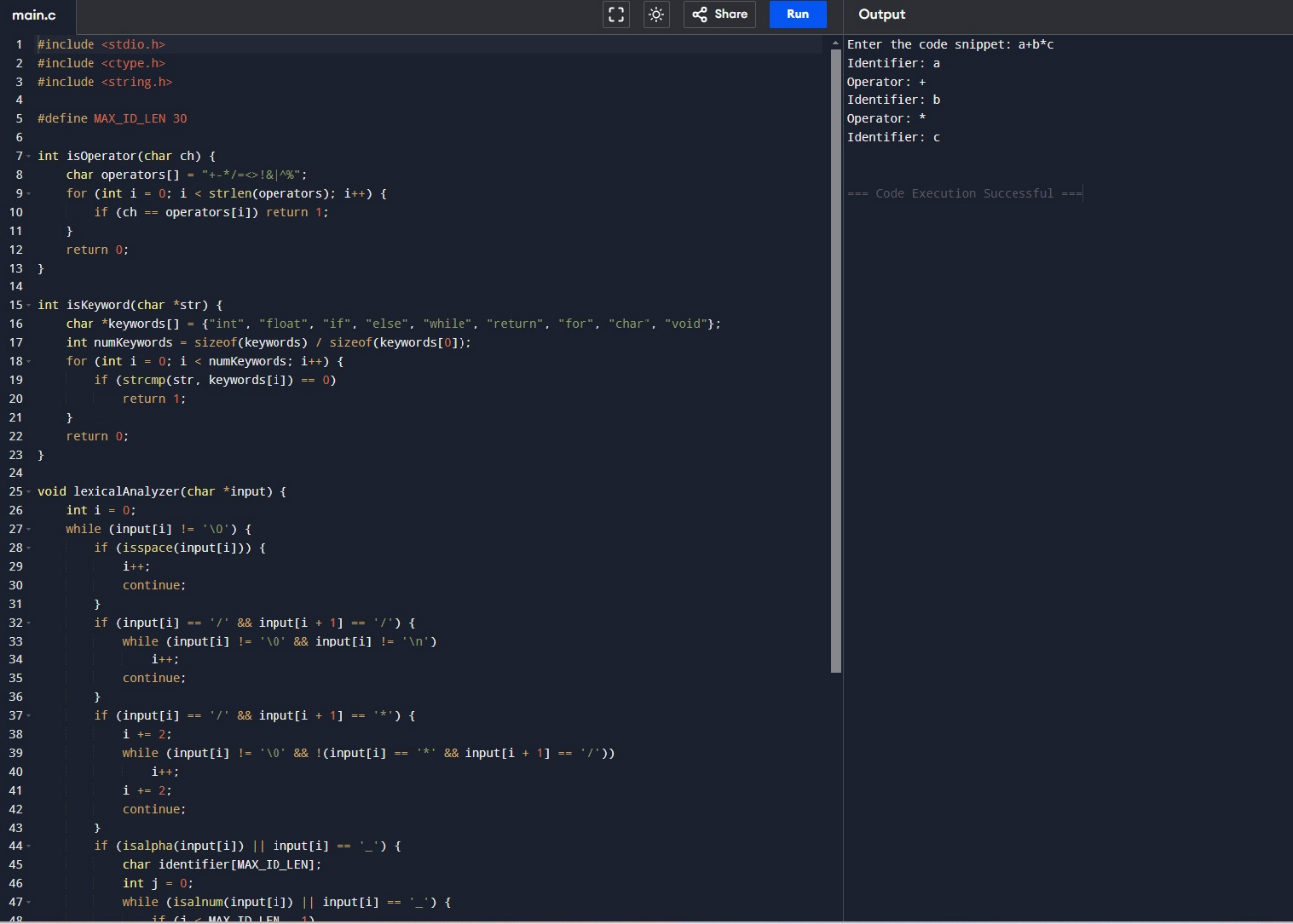
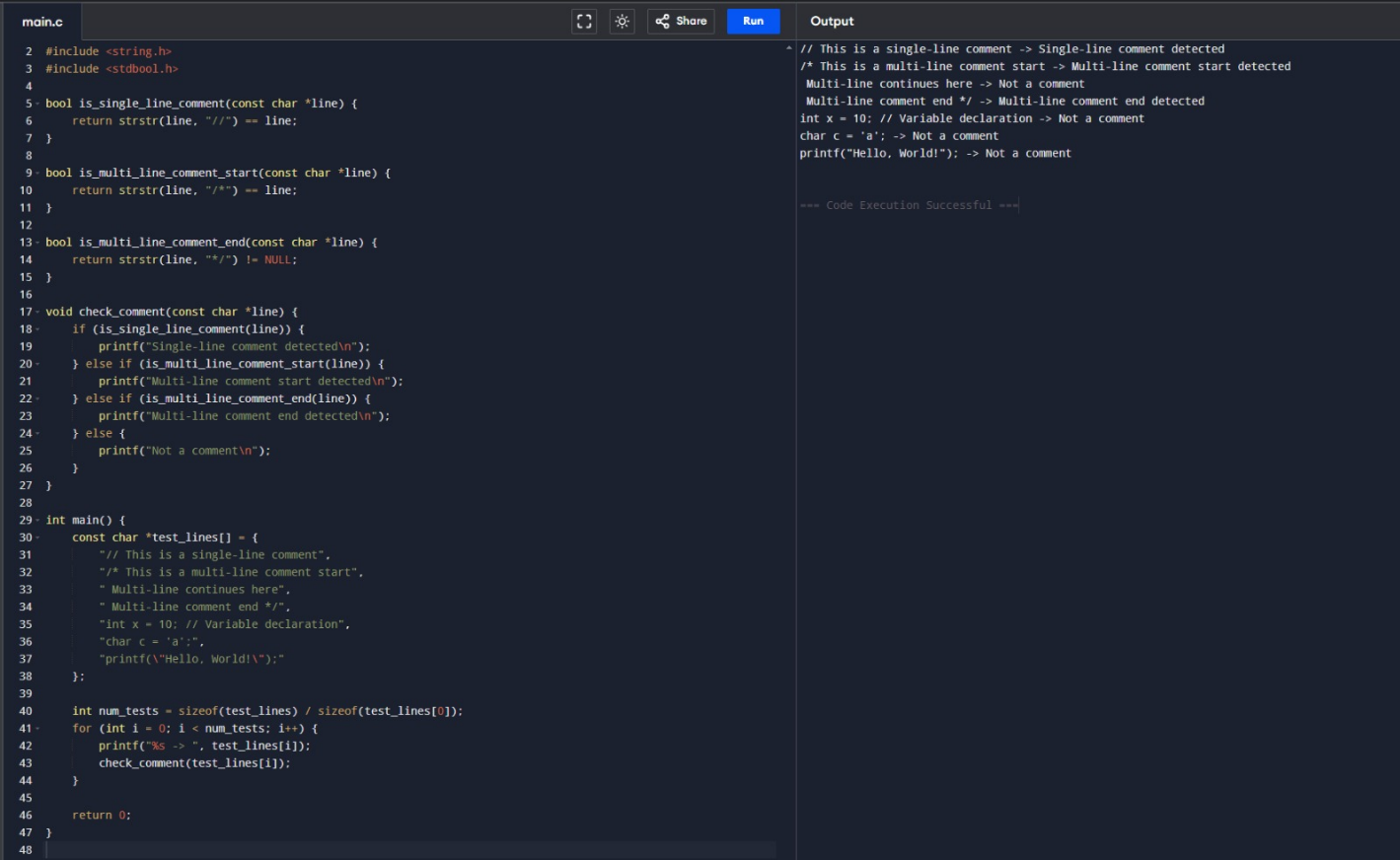
**LIST OF EXPERIMENTS**

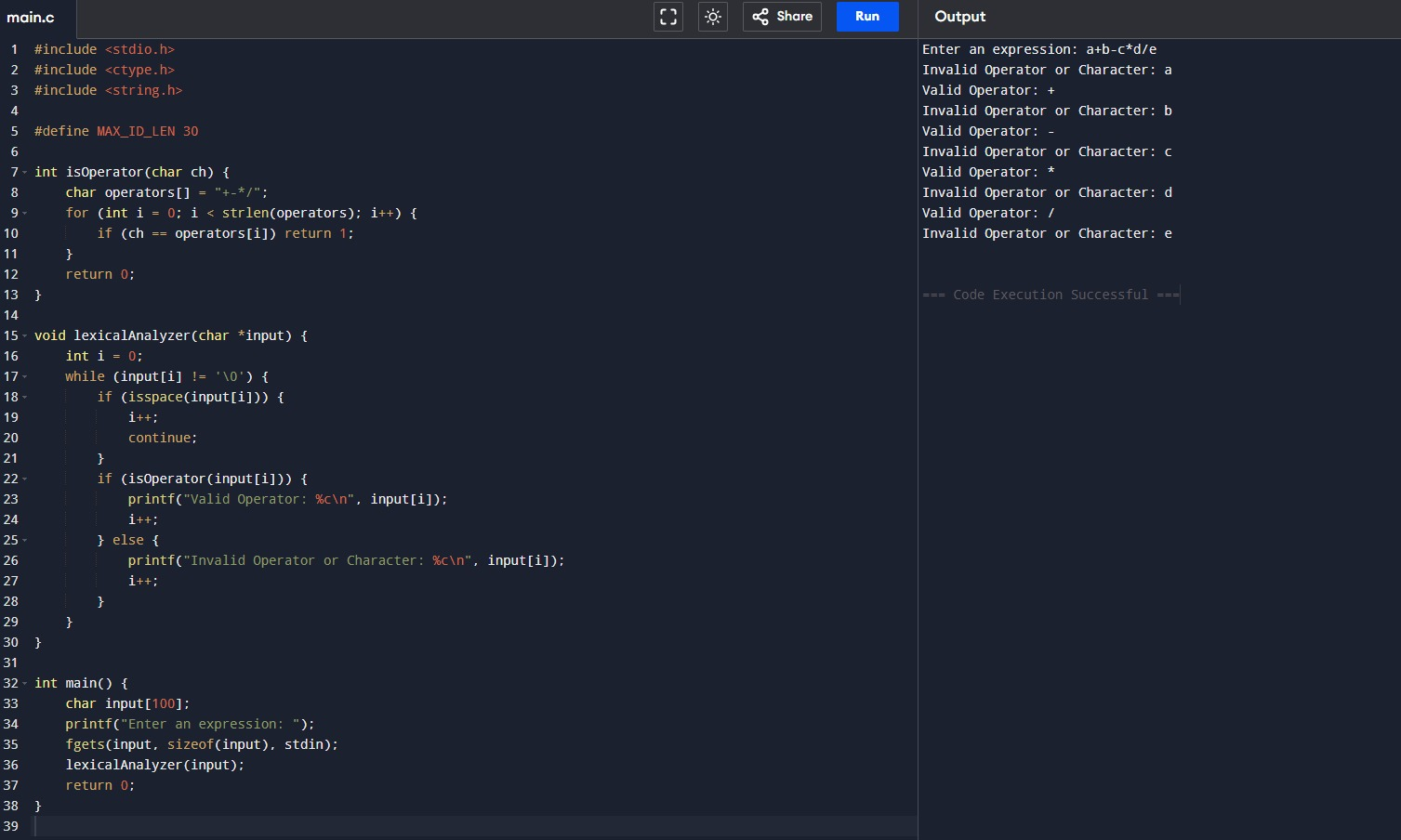
1. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Develop a lexical Analyzer to identify identifiers, constants, operators using C program.



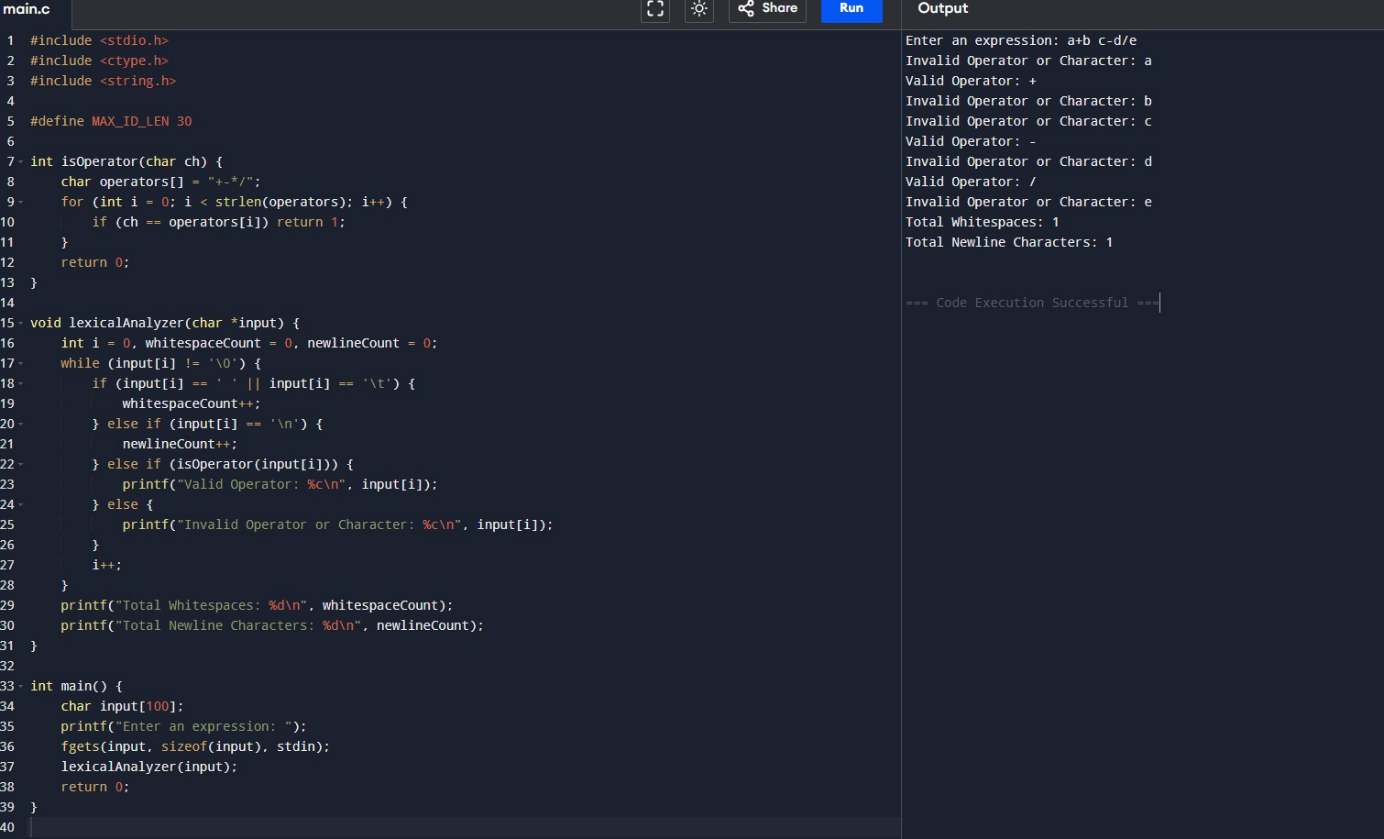
1. Extend the lexical Analyzer to Check comments, dened as follows in C:
   1. A comment begins with // and includes all characters until the end of that line.
   2. A comment begins with /\* and includes all characters through the next occurrence of the character sequence \*/Develop a lexical Analyzer to identify whether a given line is a comment or not.



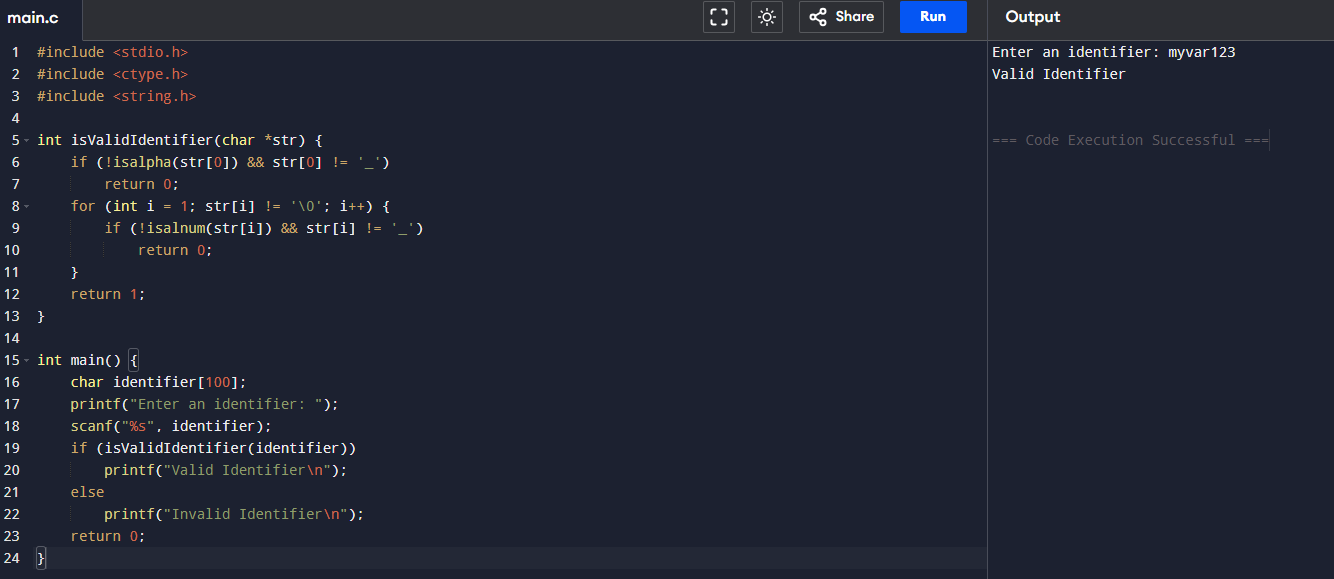
1. Design a lexical Analyzer to validate operators to recognize the operators +,-,\*,/ using regular Arithmetic operators .



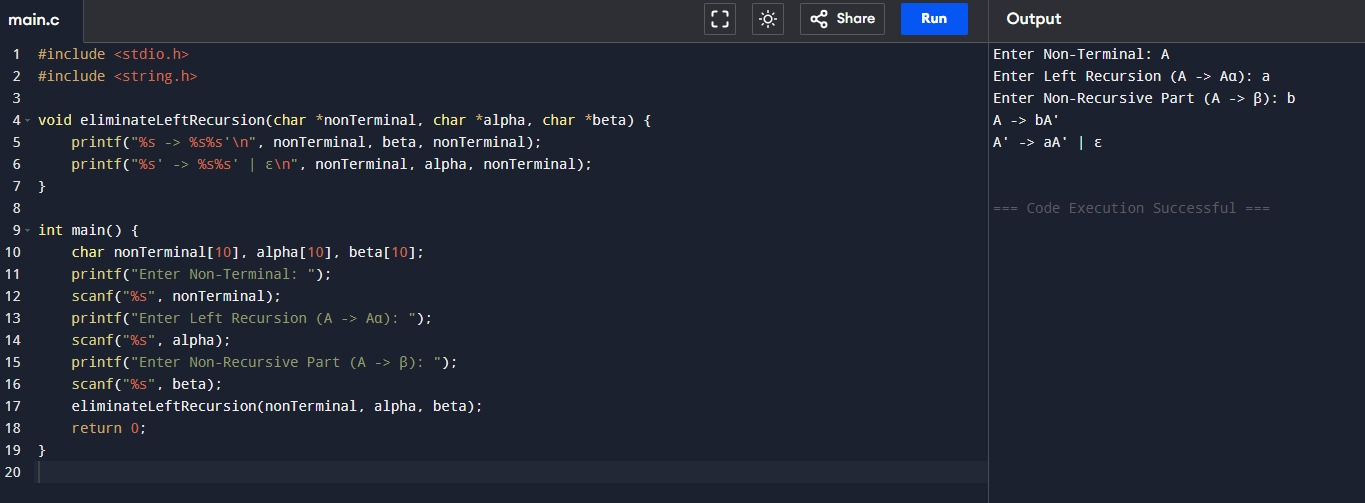
1. Design a lexical Analyzer to find the number of whitespaces and newline characters.



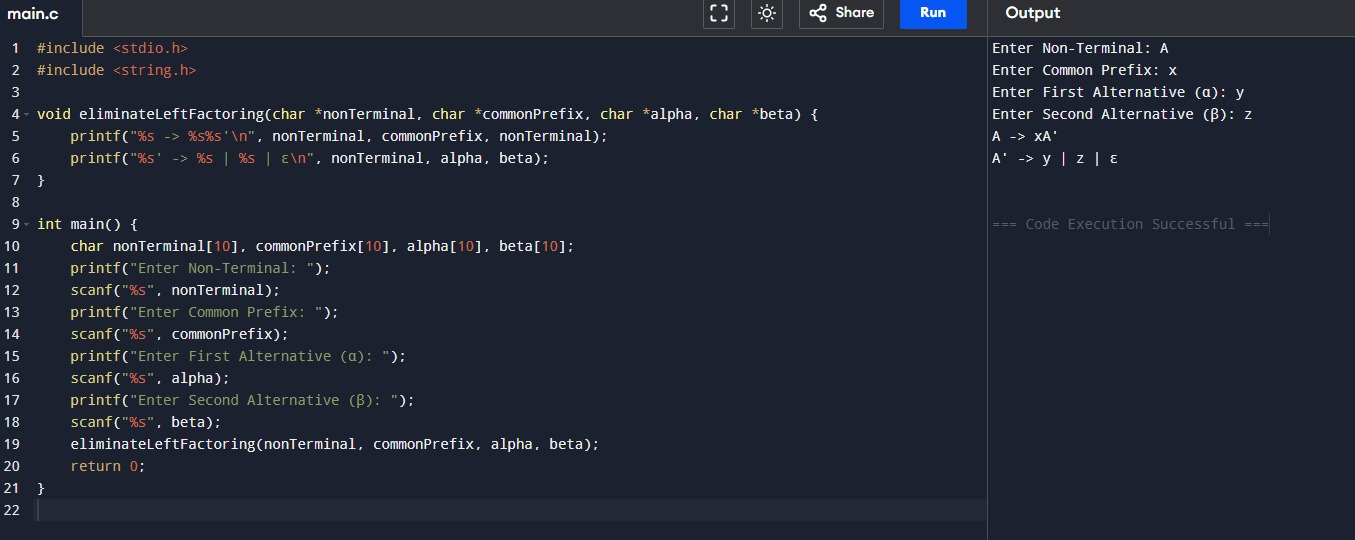
1. Develop a lexical Analyzer to test whether a given identifier is valid or not.

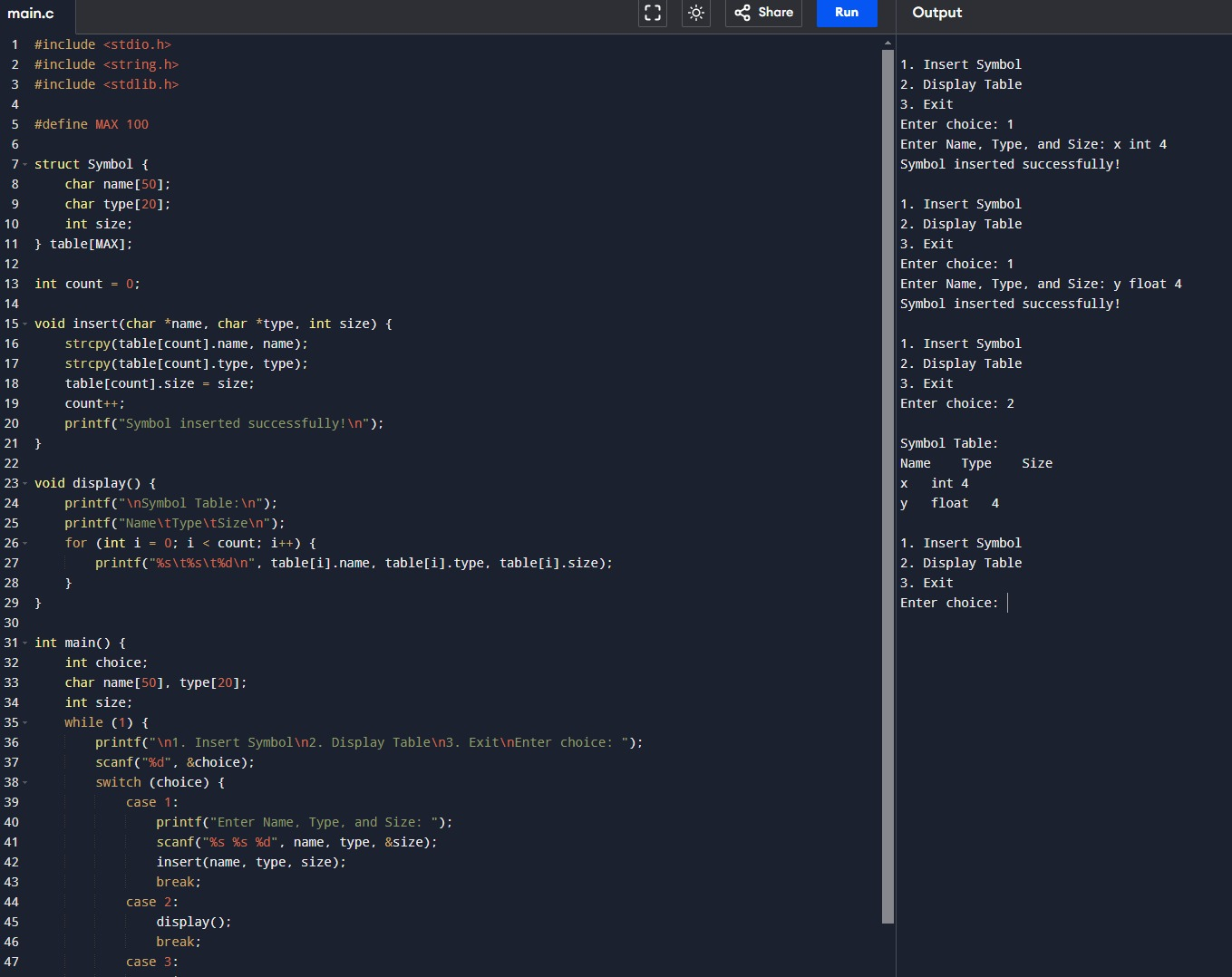


6.Implement a C program to eliminate left recursion

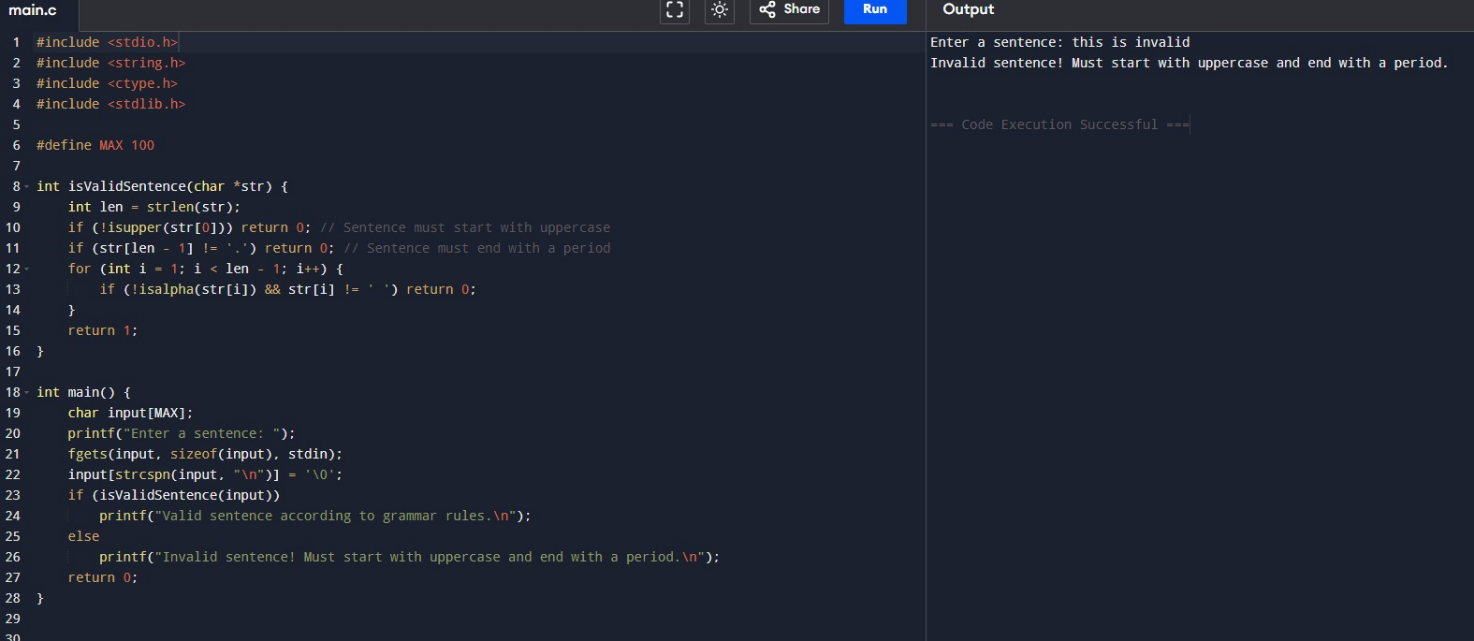


7.Implement a C program to eliminate left factoring.

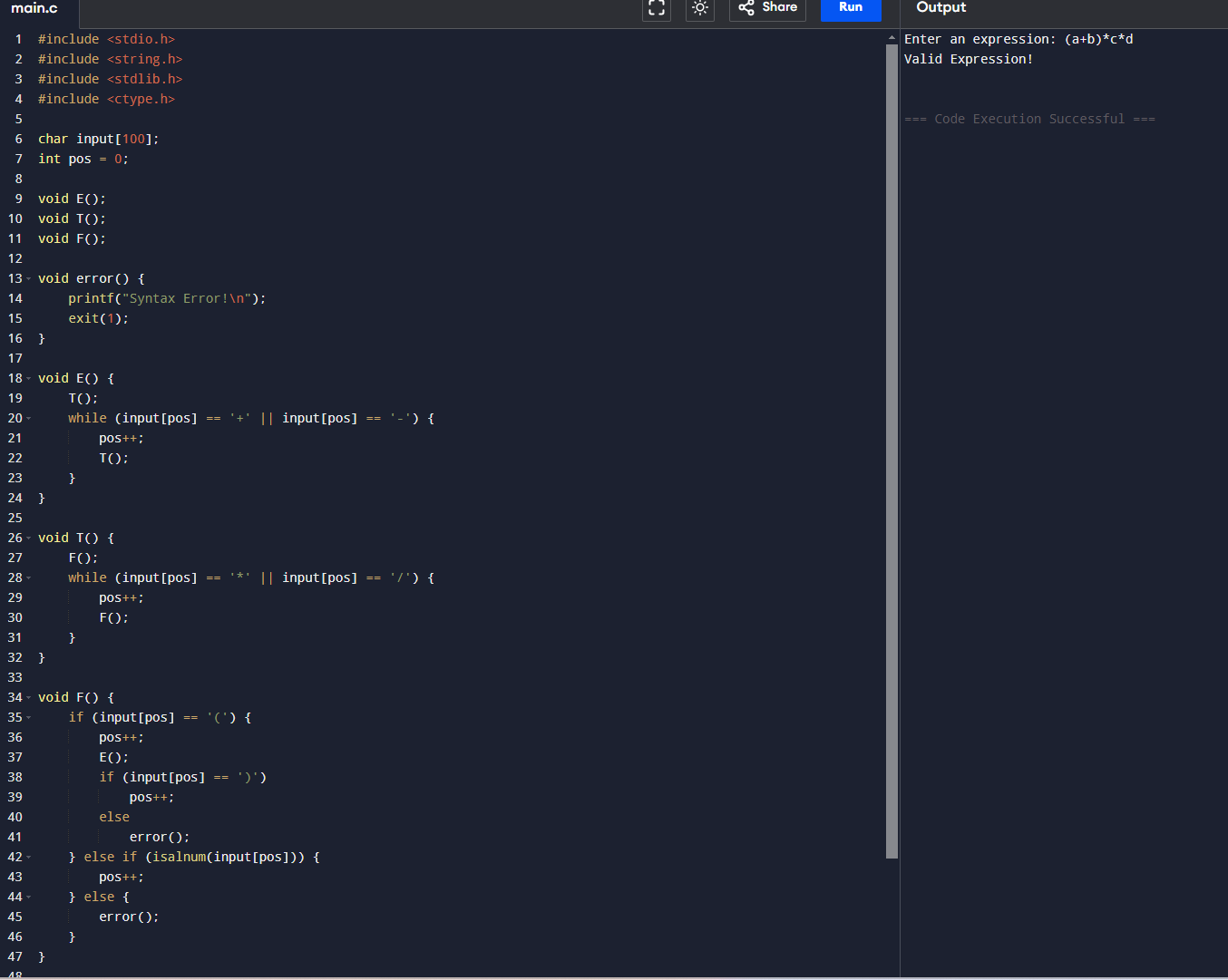


8.Implement a C program to perform symbol table operations. 

9.All languages have Grammar. When people frame a sentence we usually say whether the sentence is framed as per the rules of the Grammar or Not. Similarly use the same ideology , implement to check whether the given input string is satisfying the grammar or not .



10.Write a C program to construct recursive descent parsing.



11. In a class of Grade 3, Mathematics Teacher asked for the Acronym PEMDAS?. All of them

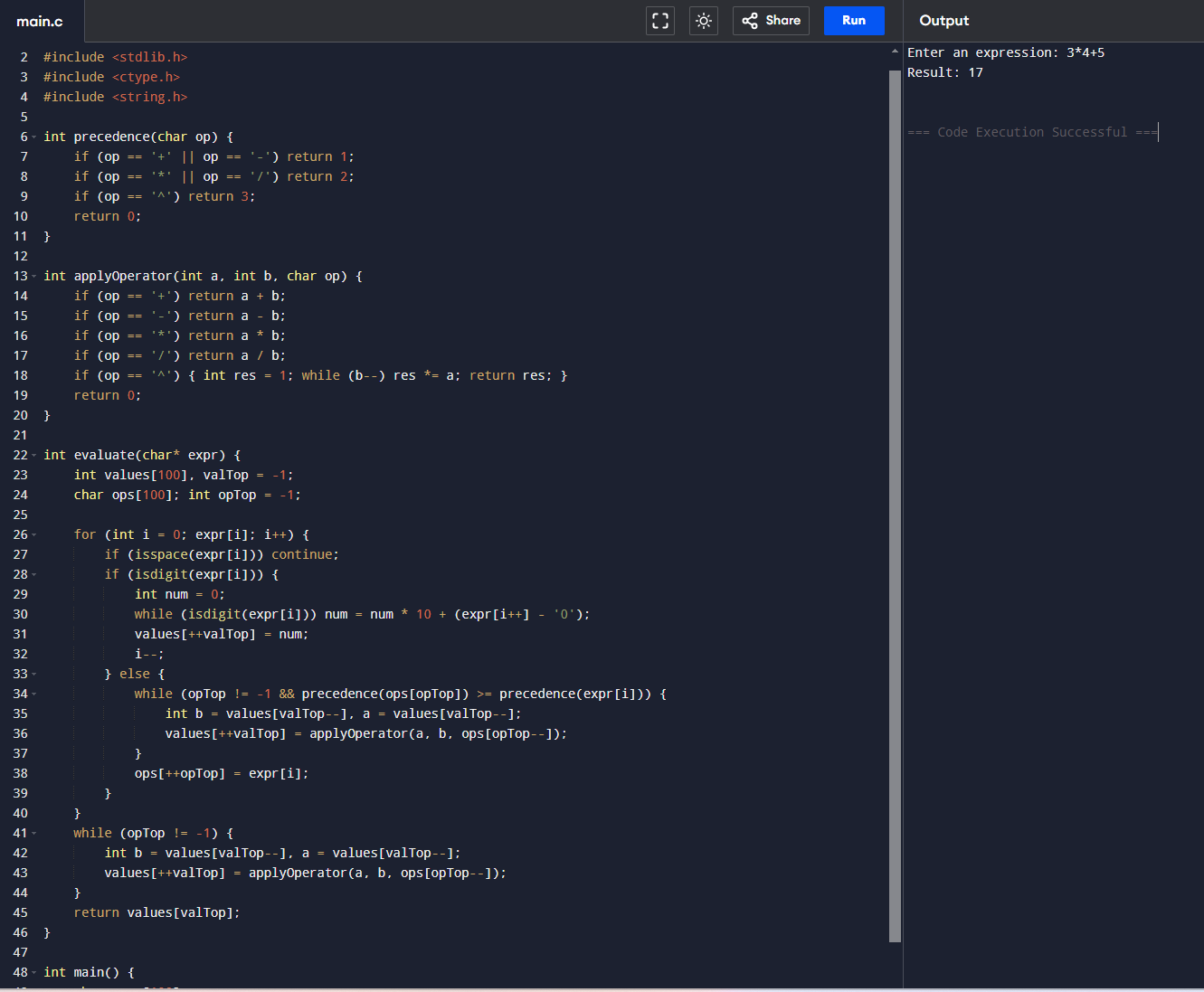
are thinking for a while. A smart kid of the class Kishore of the class says it is Parentheses,

Exponentiation, Multiplication, Division, Addition, Subtraction. Can you write a C Program

to help the students to understand about the operator precedence parsing for an expression

containing more than one operator, the order of evaluation depends on the order of

operations



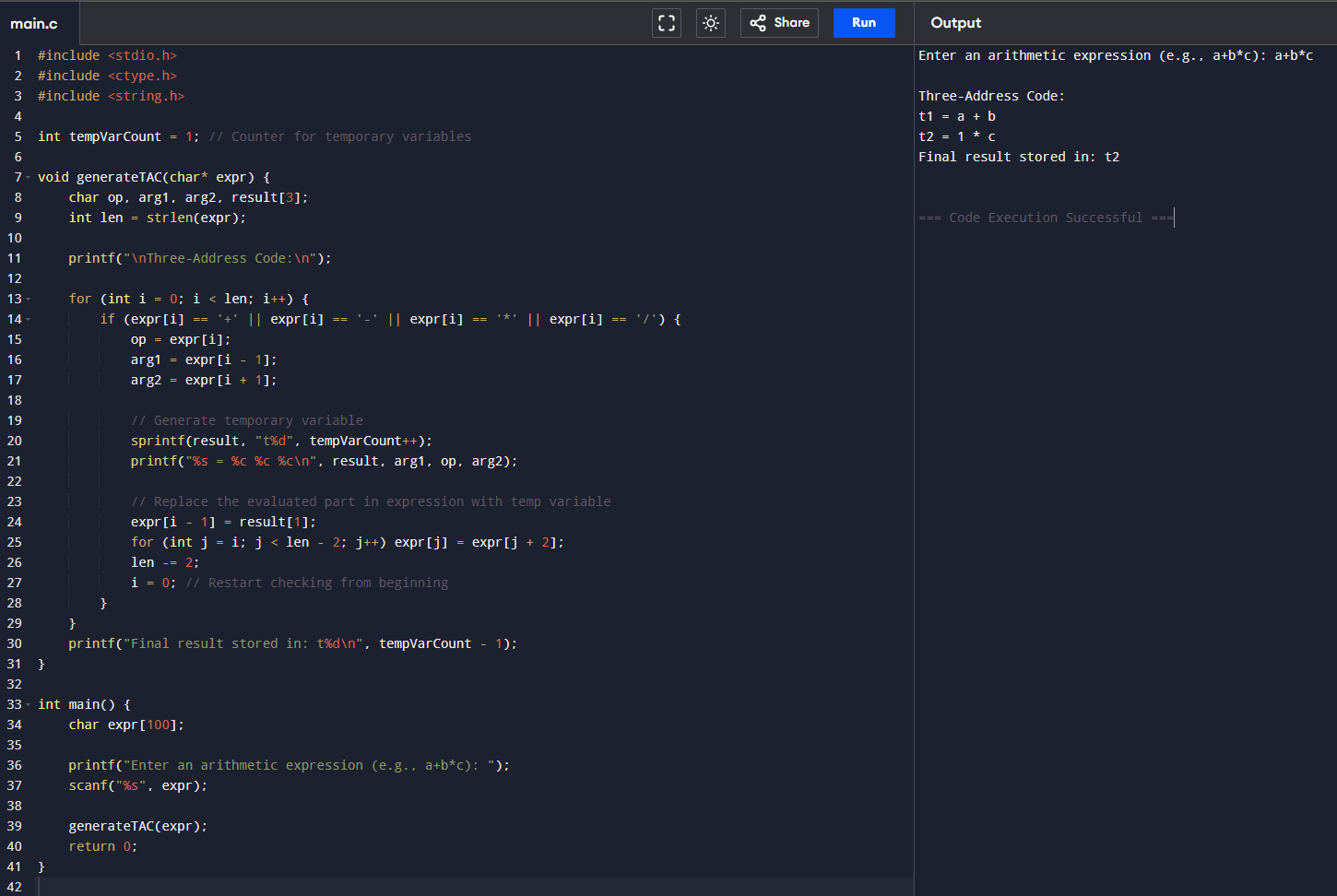
12. The main function of the Intermediate code generation is producing three address code

statements for a given input expression. The three address codes help in determining the

sequence in which operations are actioned by the compiler. The key work of Intermediate

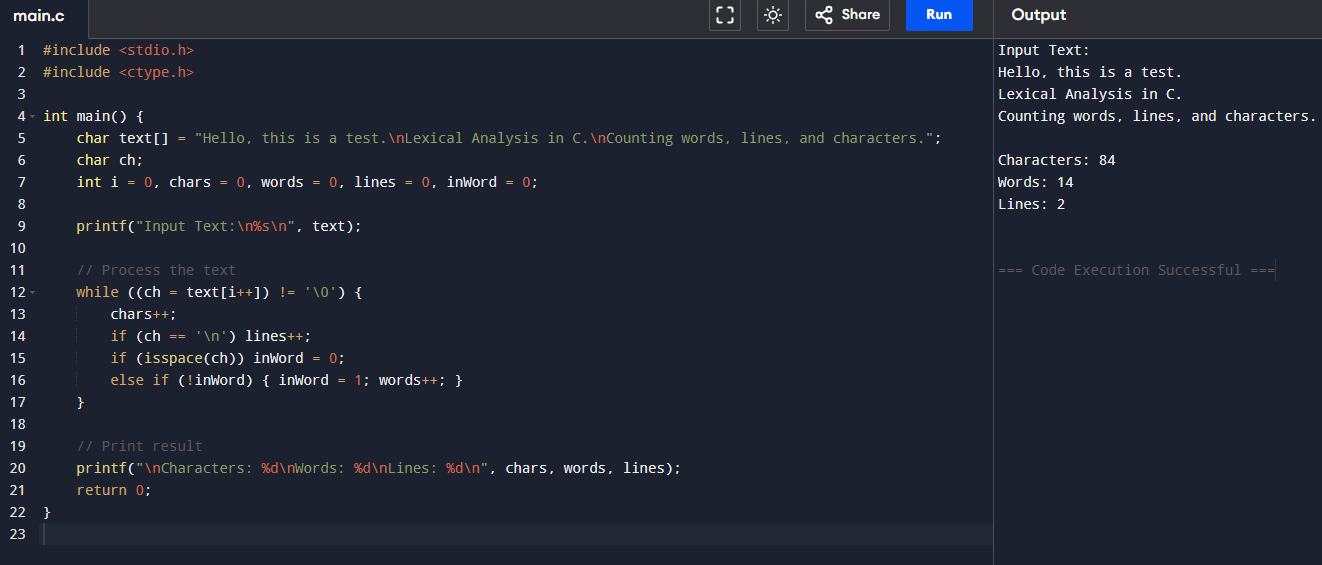
code generators is to simplify the process of Code Generator. Write a C Program to

Generate the Three address code representation for the given input statement.

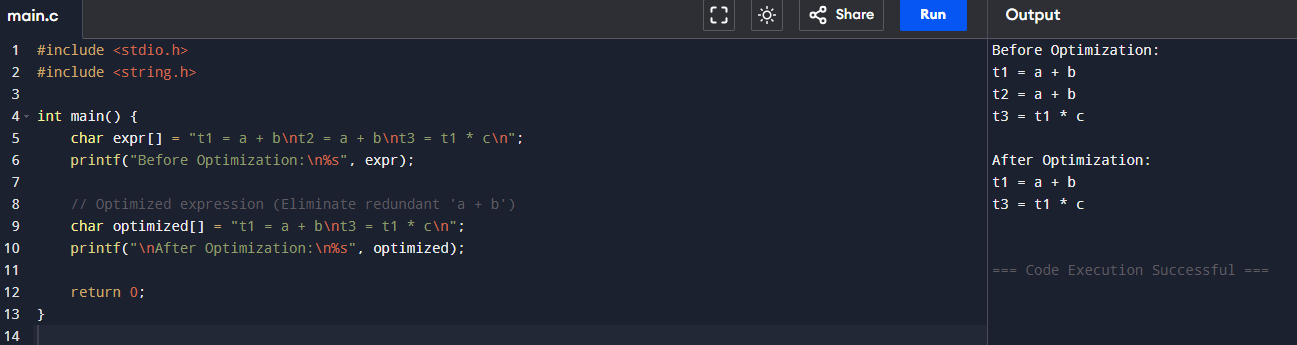


13. Write a C program for implementing a Lexical Analyzer to Count the number of characters,

words, and lines



14. Write a C Program for code optimization to eliminate common subexpression.



15. Write a C program to implement the back end of the compiler.

