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

#include <stdio.h>

#include <ctype.h>

int main() {

FILE \*fp;

char filename[100], ch;

int characters = 0, words = 0, lines = 0, inWord = 0;

printf("Enter the file name: ");

scanf("%s", filename);

fp = fopen(filename, "r");

if (fp == NULL) {

printf("File not found!\n");

return 0;

}

while ((ch = fgetc(fp)) != EOF) {

characters++;

if (ch == '\n')

lines++;

if (isspace(ch))

inWord = 0;

else if (!inWord) {

inWord = 1;

words++;

}

}

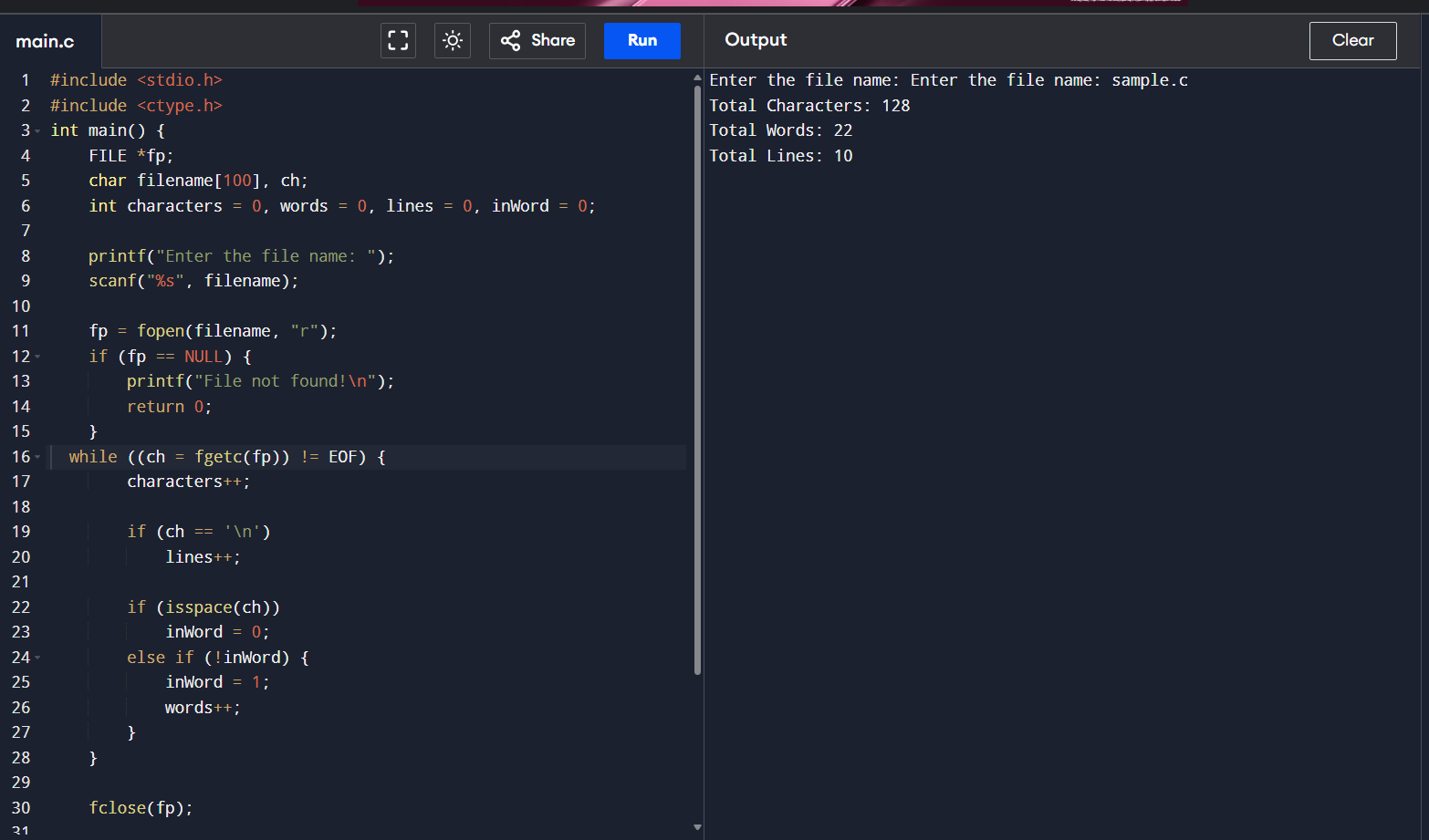
fclose(fp);

printf("\nTotal Characters: %d", characters);

printf("\nTotal Words: %d", words);

printf("\nTotal Lines: %d\n", lines);

return 0;

}

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

#include <stdio.h>

#include <string.h>

struct expr {

char lhs[10], op1[10], op[3], op2[10];

};

int main() {

int n, i, j, k = 0;

struct expr e[10];

printf("Enter number of expressions: ");

scanf("%d", &n);

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

printf("Enter expression %d (format: result operand1 operator operand2): ", i + 1);

scanf("%s %s %s %s", e[i].lhs, e[i].op1, e[i].op, e[i].op2);

}

printf("\nOptimized Code:\n");

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

int redundant = 0;

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

if (strcmp(e[i].op1, e[j].op1) == 0 &&

strcmp(e[i].op2, e[j].op2) == 0 &&

strcmp(e[i].op, e[j].op) == 0) {

redundant = 1;

printf("%s = %s\n", e[i].lhs, e[j].lhs);

break;

}

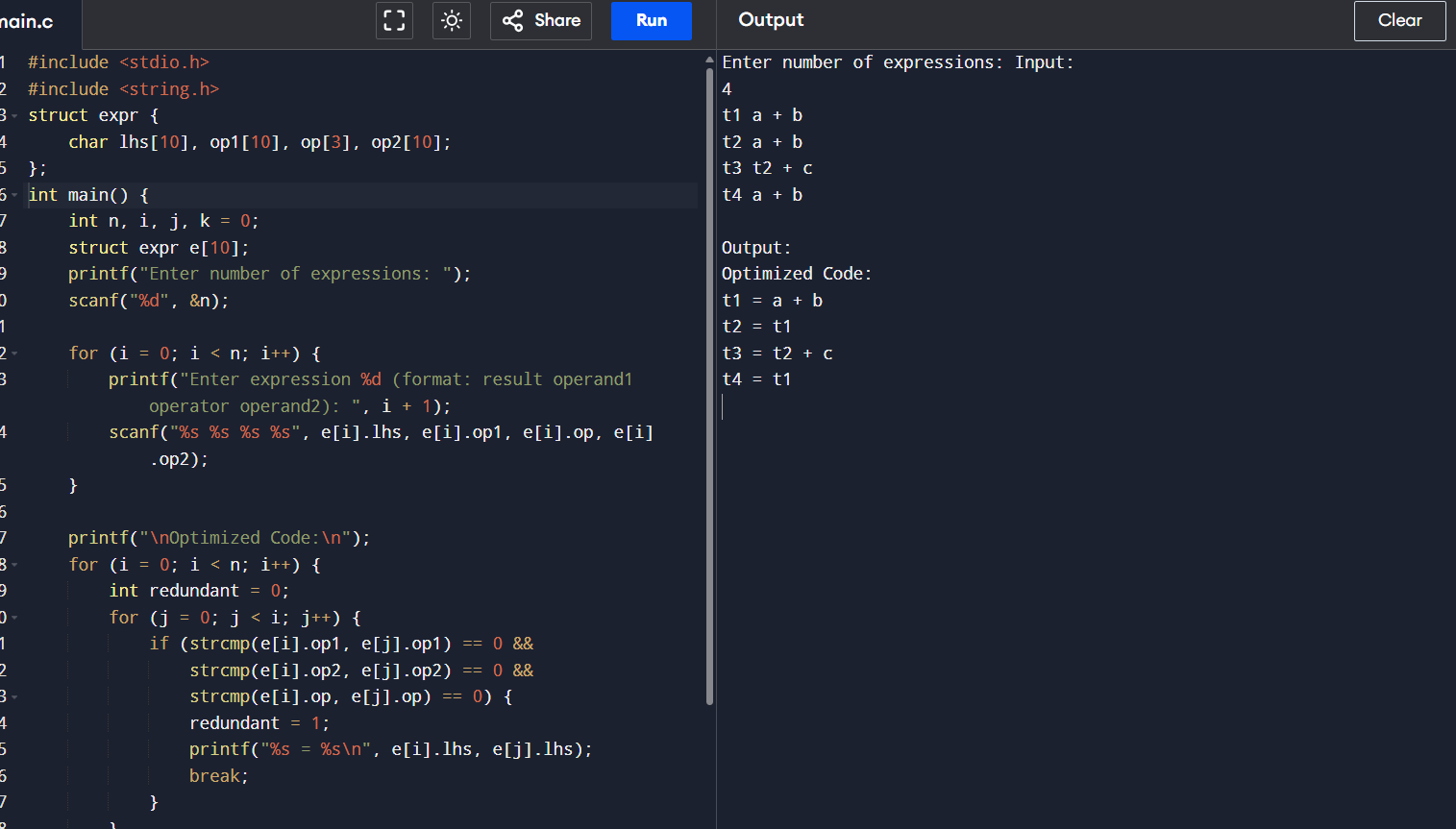
}

if (!redundant)

printf("%s = %s %s %s\n", e[i].lhs, e[i].op1, e[i].op, e[i].op2);

}

return 0;

}

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

#include <stdio.h>

#include <string.h>

int main() {

char op[5], arg1[5], arg2[5], result[5];

int n;

printf("Enter number of three address statements: ");

scanf("%d", &n);

for (int i = 0; i < n; i++) {

printf("\nEnter statement %d (result op arg1 arg2): ", i + 1);

scanf("%s %s %s %s", result, op, arg1, arg2);

printf("MOV R0, %s\n", arg1);

if (strcmp(op, "+") == 0)

printf("ADD R0, %s\n", arg2);

else if (strcmp(op, "-") == 0)

printf("SUB R0, %s\n", arg2);

else if (strcmp(op, "\*") == 0)

printf("MUL R0, %s\n", arg2);

else if (strcmp(op, "/") == 0)

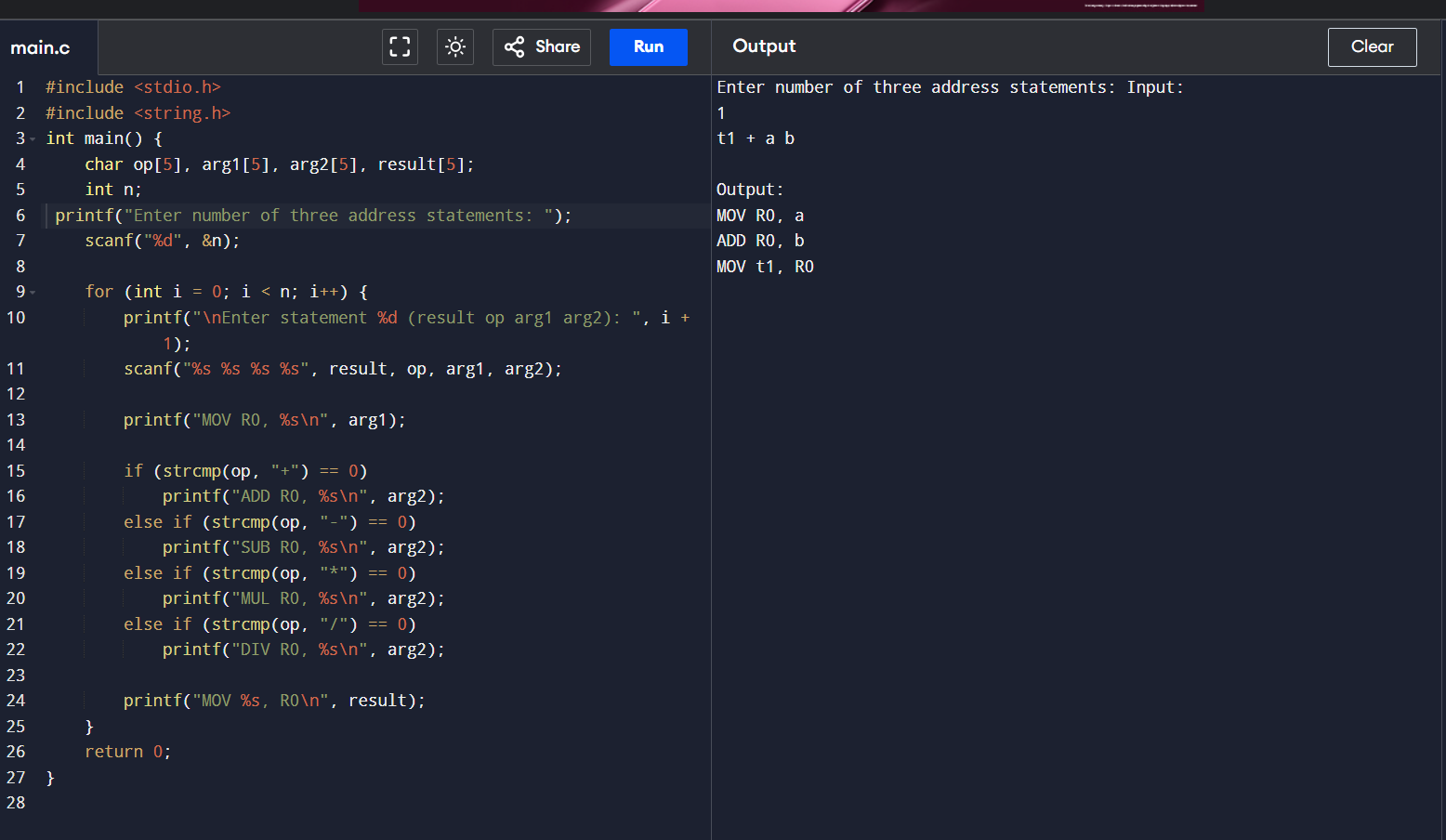
printf("DIV R0, %s\n", arg2);

printf("MOV %s, R0\n", result);

}

return 0;

}



16. 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. Write a LEX specification file to take input C program from a .c file and count the number of characters, number of lines & number of words.

#include <stdio.h>

int main()

{

int number1, number2,

sum;

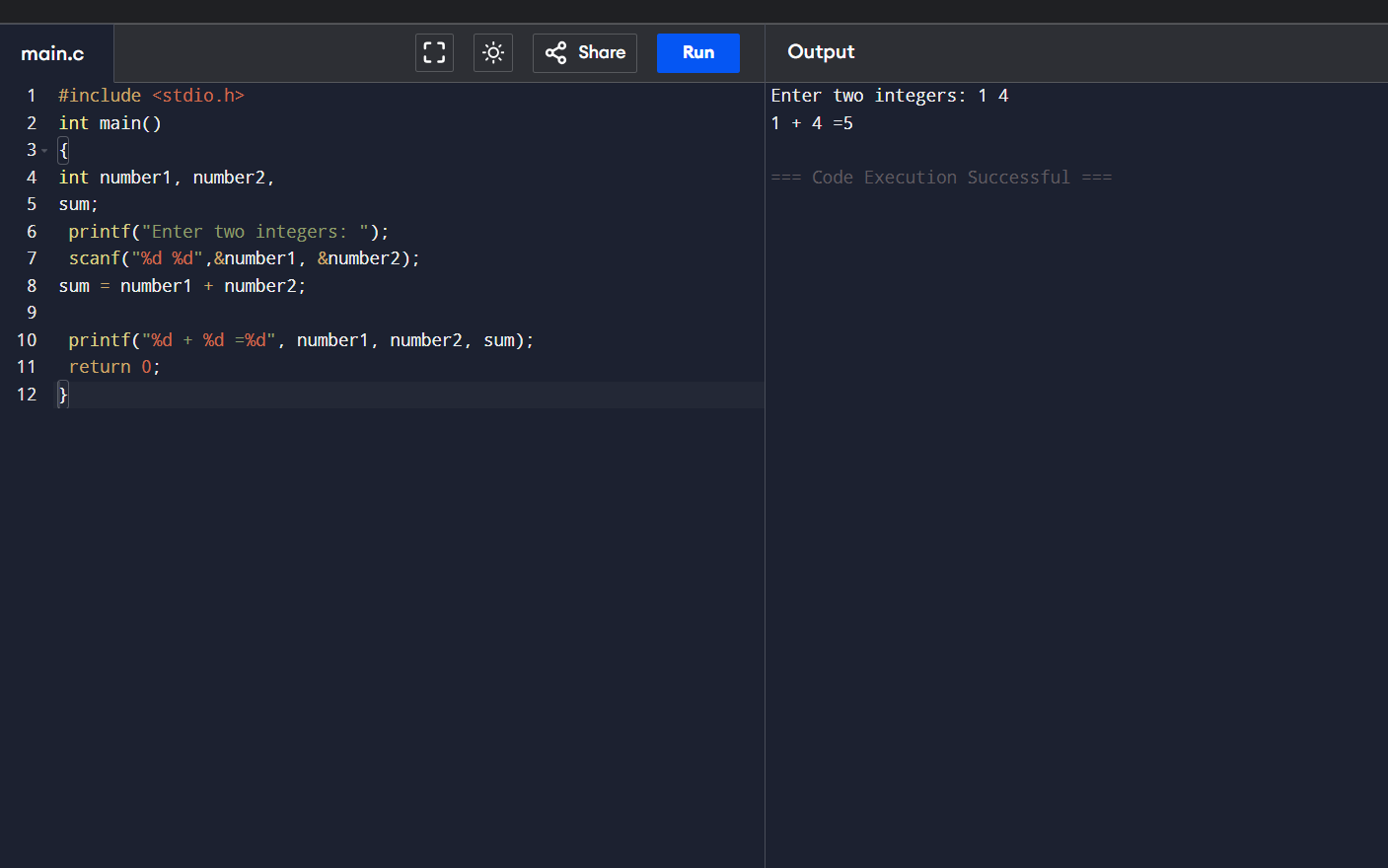
printf("Enter two integers: ");

scanf("%d %d",&number1, &number2);

sum = number1 + number2;

printf("%d + %d =%d", number1, number2, sum);

return 0;

}

17. Write a LEX program to print all the constants in the given C source program file.

#include<stdio.h> #include<conio.h>  
  
  
 void main()  
  
  
{  
  
  
            int a,b,c = 30;  
  
  
printf("hello");  
  
  
}

