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.

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_ID\_LEN 31

int isOperator(char ch) {

return (ch == '+' || ch == '-' || ch == '\*' || ch == '/');

}

int isIdentifier(char \*str) {

if (!isalpha(str[0]) && str[0] != '\_') return 0;

for (inti = 1; str[i] != '\0'; i++) {

if (!isalnum(str[i]) && str[i] != '\_') return 0;

}

return 1;

}

int main() {

char ch, buffer[100];

int i = 0;

printf("Enter the input (press Ctrl+D or Ctrl+Z to end input):\n");

while ((ch = getchar()) != EOF) {

// Skip whitespace

if (isspace(ch))

continue;

// Skip comments

if (ch == '/') {

char next = getchar();

if (next == '/') {

while ((ch = getchar()) != '\n' && ch != EOF);

} else if (next == '\*') {

while (1) {

ch = getchar()

if (ch == '\*') {

if ((ch = getchar()) == '/') break;

}

if (ch == EOF) break;

}

} else {

// It's a division operator

printf("Operator: /\n");

ungetc(next, stdin); // put back the character

}

} else if (isOperator(ch)) {

printf("Operator: %c\n", ch);

} else if (is digit(ch)) {

buffer[0] = ch;

i = 1;

while (isdigit(ch = getchar())) {

buffer[i++] = ch;

}

buffer[i] = '\0';

printf("Constant: %s\n", buffer);

ungetc(ch, stdin);

} else if (isalpha(ch) || ch == '\_') {

buffer[0] = ch;

i = 1;

while (isalnum(ch = getchar()) || ch == '\_') {

if (i < MAX\_ID\_LEN)

buffer[i++] = ch;

}

buffer[i] = '\0';

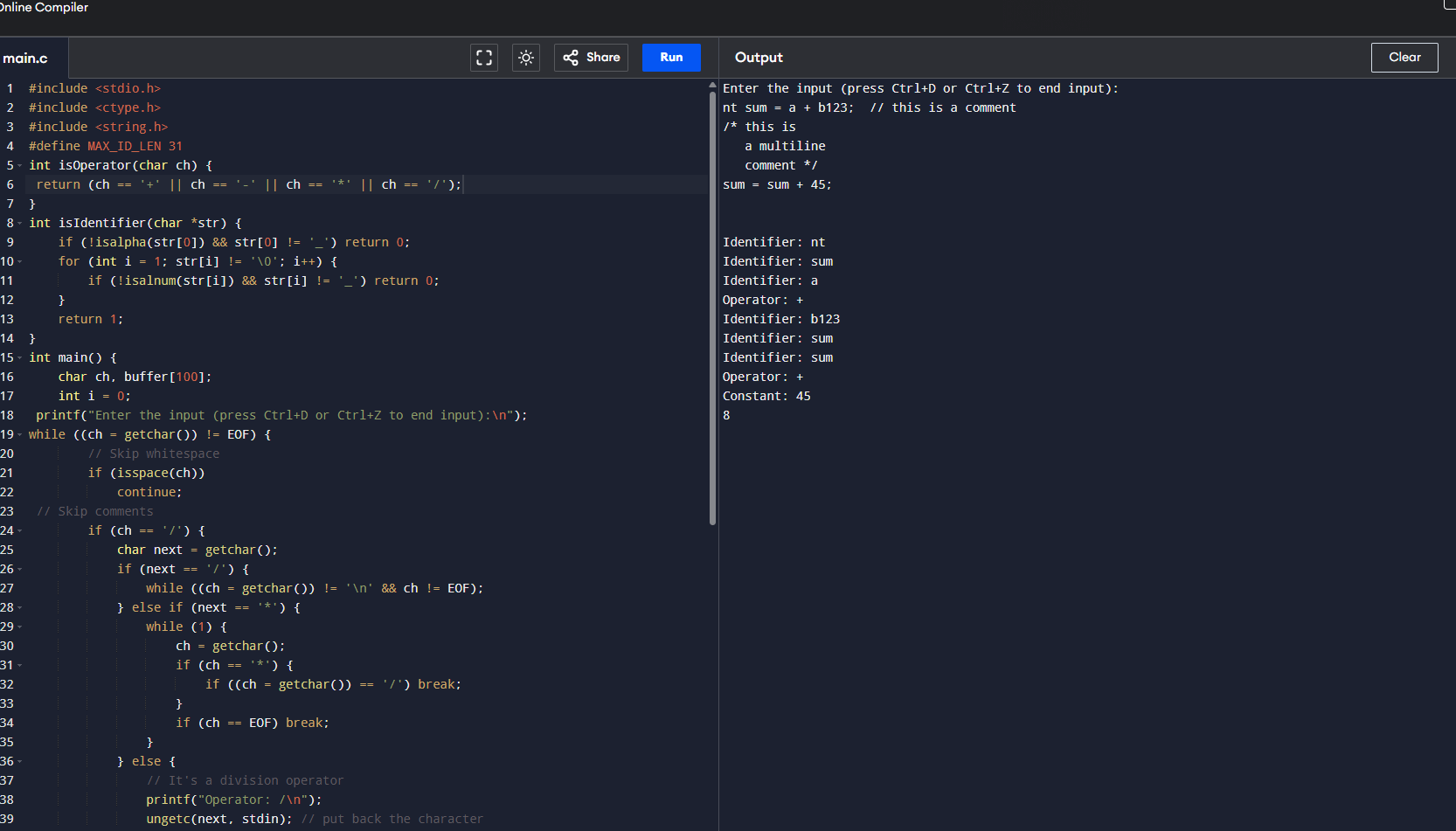
printf("Identifier: %s\n", buffer);

ungetc(ch, stdin)

}

}

return 0;

}

2.     Extend the lexical Analyzer to Check comments, dened as follows in C:

 a) A comment begins with // and includes all characters until the end of that line.  
 b) 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.

#include <stdio.h>

#include <string.h>

int main() {

char line[256];

printf("Enter a line of code:\n");

fgets(line, sizeof(line), stdin);

if (strstr(line, "//")) {

printf("This is a single-line comment.\n");

} else if (strstr(line, "/\*") && strstr(line, "\*/")) {

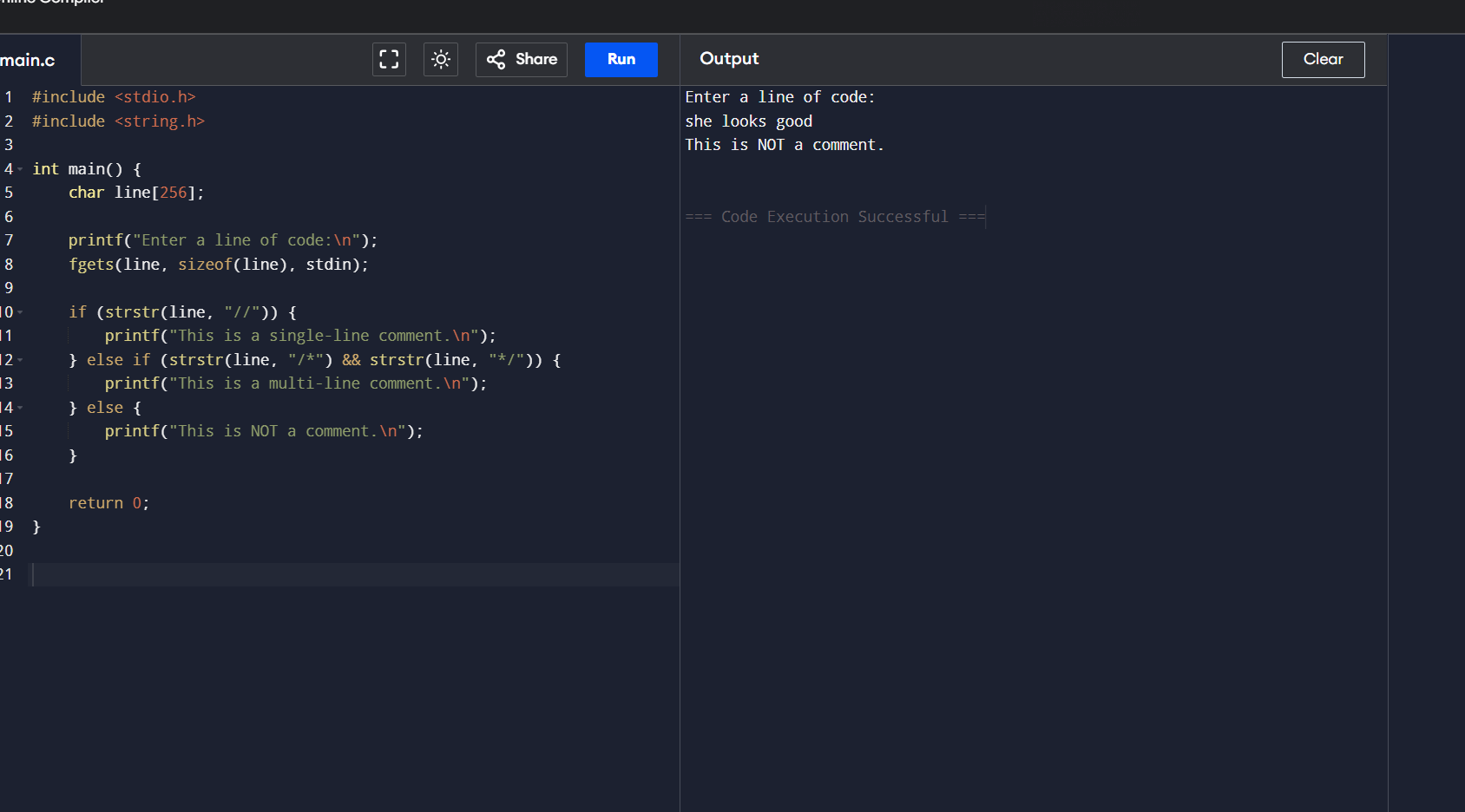
printf("This is a multi-line comment.\n");

} else {

printf("This is NOT a comment.\n");

}

return 0;

}

3. Design

a lexical

Analyzer to validate operators to

recognize the operators +,-,\*,/ using regular Arithmetic operators .

#include <stdio.h>

int main() {

char op;

printf("Enter an operator: ");

scanf(" %c", &op); // space before %c to consume any whitespace

if (op == '+' || op == '-' || op == '\*' || op == '/') {

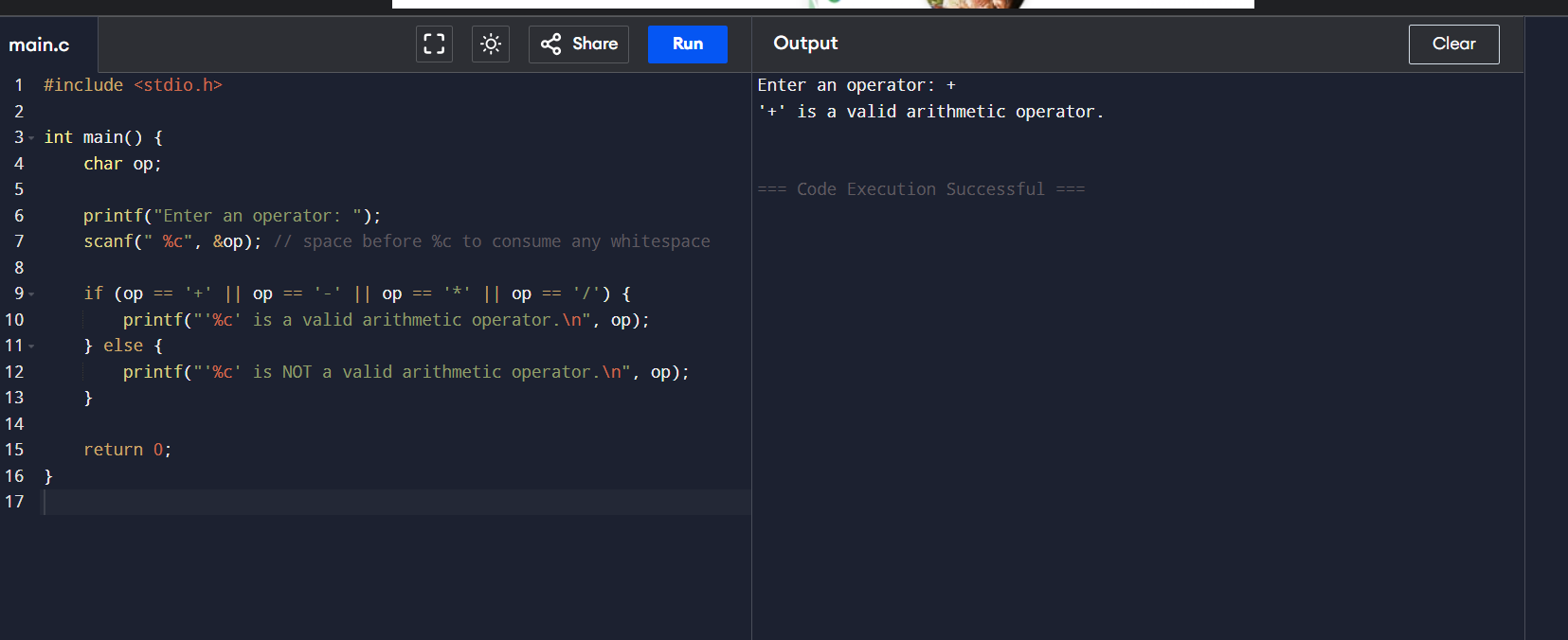
printf("'%c' is a valid arithmetic operator.\n", op);

} else {

printf("'%c' is NOT a valid arithmetic operator.\n", op);

}

return 0;

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

#include <stdio.h>

int main() {

char ch;

int spaceCount = 0, newlineCount = 0;

printf("Enter text (Press Ctrl+D to end input on Unix/Linux or Ctrl+Z on Windows):\n");

while ((ch = getchar()) != EOF) {

if (ch == ' ')

spaceCount++;

else if (ch == '\n')

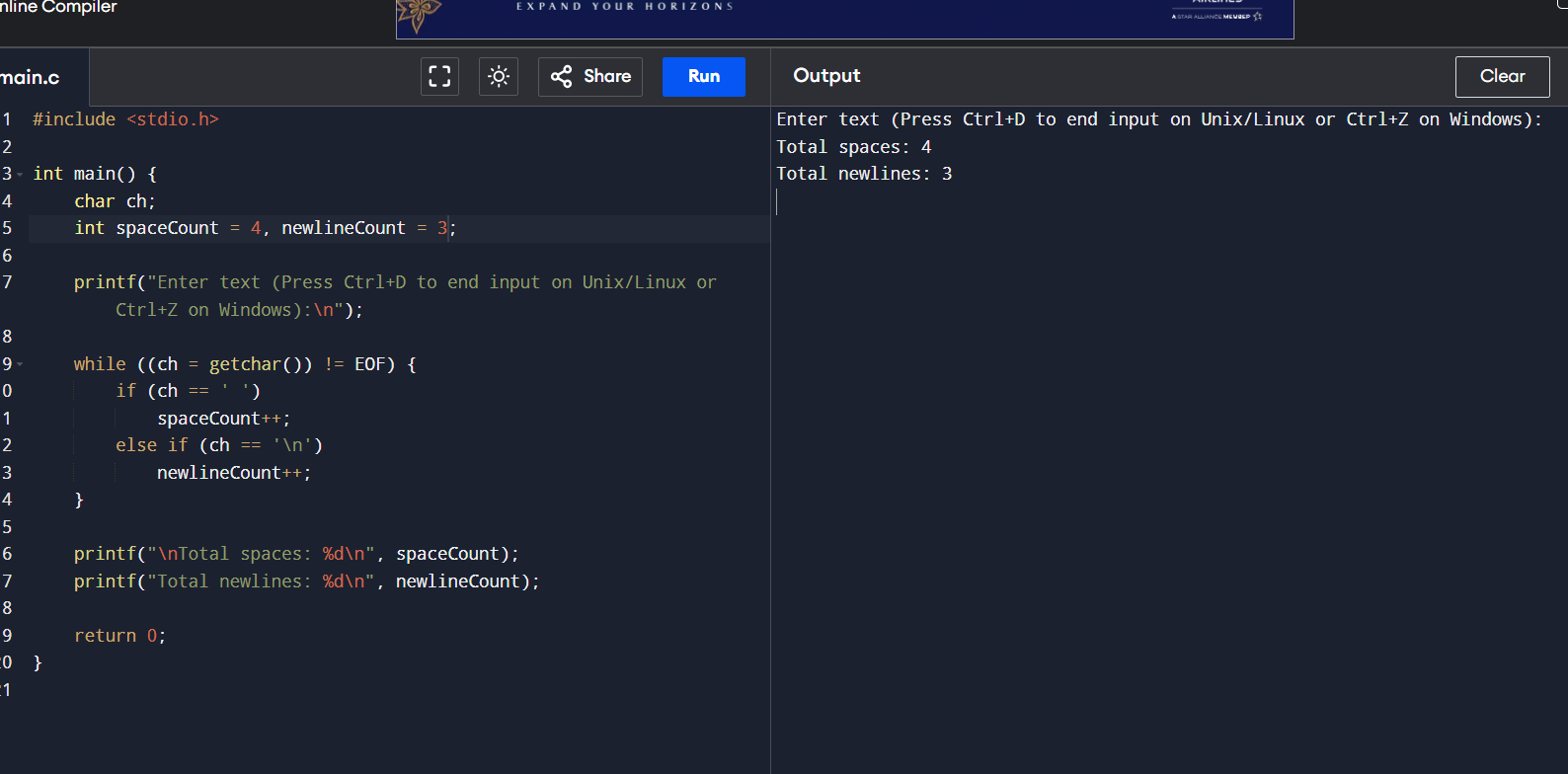
newline Count++;

}

printf("\nTotal spaces: %d\n", spaceCount);

printf("Total newlines: %d\n", newlineCount);

return 0;

}

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

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int is Key word(char str[]) {

// A small list of C keywords

char\* keywords[] = {

"int", "float", "return", "if", "else", "while", "for", "char", "double", "void"

};

int n = sizeof(keywords) / sizeof(keywords[0]);

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

if (strcmp(str, keywords[i]) == 0)

return 1;

}

return 0;

}

int isValidIdentifier(char str[]) {

if (!(isalpha(str[0]) || str[0] == '\_'))

return 0;

for (int i = 1; str[i] != '\0'; i++) {

if (!(isalnum(str[i]) || str[i] == '\_'))

return 0;

}

return !isKeyword(str);

}

int main() {

char identifier[100];

printf("Enter an identifier: ");

scanf("%s", identifier);

if (isKeyword(identifier)) {

printf("'%s' is a keyword, not a valid identifier.\n", identifier);

} else if (isValidIdentifier(identifier)) {

printf("'%s' is a valid identifier.\n", identifier);

} else {

printf("'%s' is NOT a valid identifier.\n", identifier);

}

return 0;

}