CSA1413-COMPILER DESIGN FOR INTERMEDIATE LANGUAGE
LAB MANUAL

PARAMESWARI.R
192324118

PROGRAM 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

CODE:

```
   #include <stdio.h>
#include <ctype.h>
int main() {
   char c;
   printf("Enter input (Ctrl+D to end):\n");
   while ((c = getchar()) != EOF) {
      if (c==' ' || c=='\t' || c=='\n')
         continue;
      if (isalpha(c)) {
         printf("Identifier: %c\n", c);
      }
      else if (isdigit(c)) {
         printf("Constant: %c\n", c);
      }
      else if (c=='+' || c=='-' || c=='*' || c=='/' || c=='=')
         printf("Operator: %c\n", c);
   }

   return 0;
}
```

OUTPUT:

```
  Output
Enter input (Ctrl+D to end):
A*B=10
Identifier: A
Operator: *
Identifier: B
Operator: =
Constant: 1
Constant: 0
```

PROGRAM 2:

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

CODE:

```
#include <stdio.h>
int main() {
   char ch, next;
   printf("Enter C code (Press CTRL + D or CTRL + Z to stop input):\n");
   while ((ch = getchar()) != EOF) {
      if (ch == '/') {
         next = getchar();
         if (next == '/') {
            printf("Single-line comment: //");
            while ((ch = getchar()) != '\n' && ch != EOF)
               putchar(ch);
            printf("\n");
         }
         else if (next == '*') {
            printf("Multi-line comment: /*");
            while ((ch = getchar()) != EOF) {
               putchar(ch);
               if (ch == '*') {
                  next = getchar();
                  putchar(next);
                  if (next == '/')
                     break;
               }
            }
            printf("\n");
         }
         else {
         }
      }
   }

   return 0;
}
```

INPUT & OUTPUT:

```
Enter C code (Press CTRL + D or CTRL + Z to stop input):
int x = 10; // hello world

/* this is
   a test */
x++;

Single-line comment: // hello world
Multi-line comment: /* this is
   a test */
```

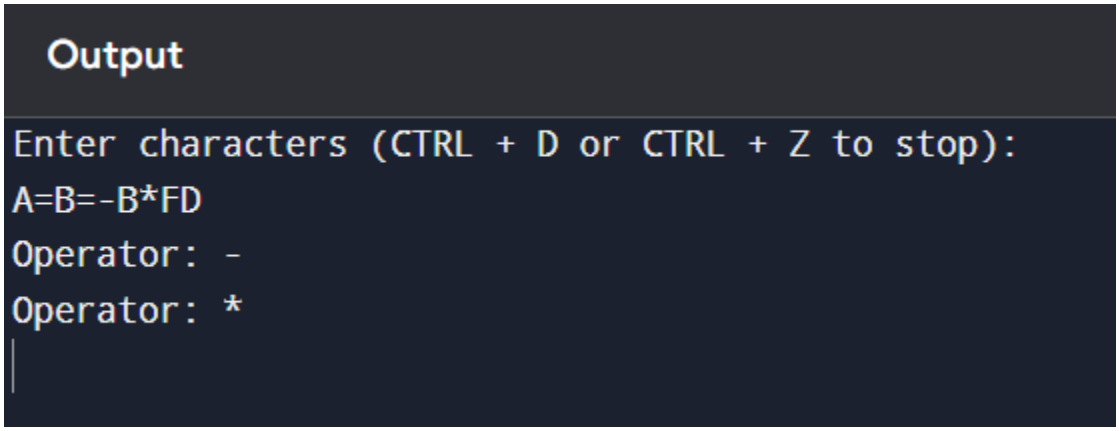
3. Design a lexical Analyzer to validate operators to recognize the operators +,-,*,/ using regular Arithmetic operators .

CODE:

```
#include <stdio.h>
int main() {
   char ch;
   printf("Enter characters (CTRL + D or CTRL + Z to stop):\n");
   while ((ch = getchar()) != EOF) {
      if (ch == '+') {
         printf("Operator: +\n");
      }
      else if (ch == '-') {
         printf("Operator: -\n");
      }
      else if (ch == '*') {
         printf("Operator: *\n");
      }
      else if (ch == '/') {
         printf("Operator: /\n");
      }
   }

   return 0;
}
```

OUTPUT:



4.PROGRAM:

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

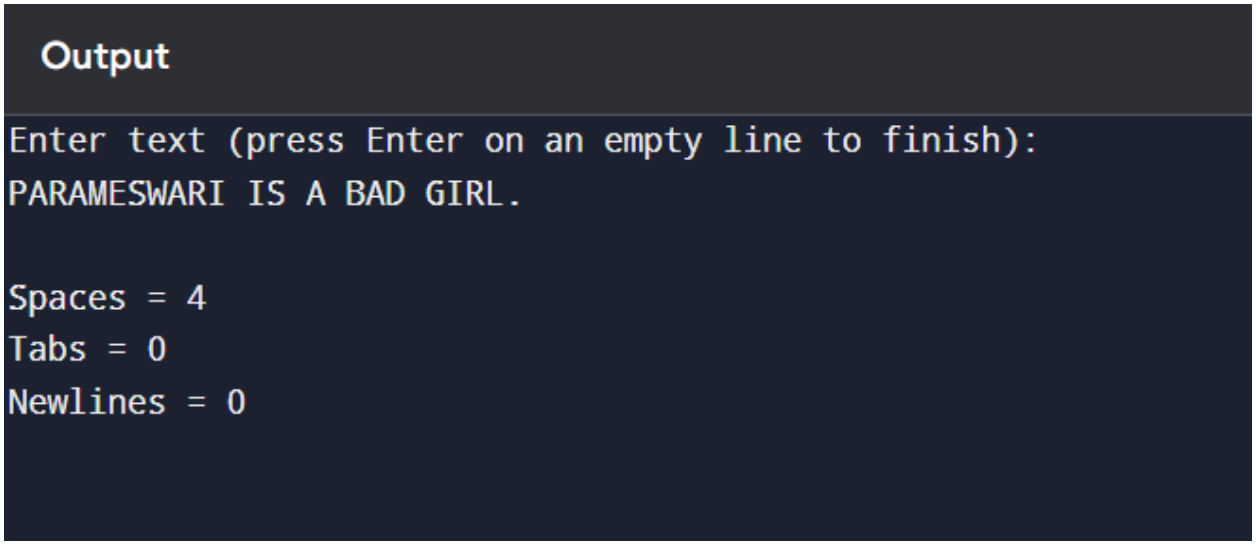
CODE:

```
#include <stdio.h>
int main() {
   char ch;
   int spaces = 0, tabs = 0, newlines = 0;
   printf("Enter text (press Enter on an empty line to finish):\n");
   while (1) {
      ch = getchar();
      if (ch == '\n') {
         newlines++;
         char next = getchar();
         if (next == '\n') break;  // stop on empty line
         ungetc(next, stdin); // put back
      }
      if (ch == ' ')
         spaces++;
      else if (ch == '\t')
         tabs++;
   }

   printf("Spaces = %d\n", spaces);
   printf("Tabs = %d\n", tabs);
   printf("Newlines = %d\n", newlines - 1);

   return 0;
}
```

OUTPUT:



5. PROGRAM:

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

CODE:

```
#include <stdio.h>
#include <ctype.h>
int main() {
   char id[100];
   int i = 0, valid = 1;
   printf("Enter an identifier: ");
   scanf("%s", id);
   if (!(isalpha(id[0]) || id[0] == '_')) {
      valid = 0;
   }
   for (i = 1; id[i] != '\0'; i++) {
      if (!(isalnum(id[i]) || id[i] == '_')) {
         valid = 0;
         break;
      }
   }
   if (valid)
      printf("Valid Identifier\n");
   else
      printf("Invalid Identifier\n");
```

```
    return 0;
}
```

OUTPUT: