LAB PROGRAMS 1-5

M.SAI HARSHITHA
192424408
CSE AI & DS
CSA1402

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 ch;
   printf("Enter a line of code: ");
   while ((ch = getchar()) != '\n') {
      if (ch == ' ' || ch == '\t')
         continue;
      if (isalpha(ch))
         printf("Identifier: %c\n", ch);
      else if (isdigit(ch))
         printf("Constant: %c\n", ch);
      else if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '=')
         printf("Operator: %c\n", ch);
      else
         printf("Symbol: %c\n", ch);
   }
   return 0;
}
```

SAMPLE OUTPUT:

Enter a line of code: int 5;

Identifier: i

Identifier: n

Identifier: t

Constant: 5

Symbol: ;

Output:

```
Enter a line of code: int 5;
Identifier: i
Identifier: n
Identifier: t
Constant: 5
Symbol: ;

--------------------------------
Process exited after 4.61 seconds with return value 0
Press any key to continue . . .
```

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.

CODE:

```
#include <stdio.h>
#include <string.h>
int main() {
   char line[100];
   printf("Enter a line: ");
   gets(line);
   if (line[0] == '/' && line[1] == '/') {
      printf("This is a single-line comment.\n");
   }
   else if (line[0] == '/' && line[1] == '*') {
      int len = strlen(line);
      if (len >= 4 && line[len - 2] == '*' && line[len - 1] == '/')
         printf("This is a multi-line comment.\n");
      else
         printf("This is the beginning of a multi-line comment (not closed).\n");
   } else {
      printf("This is not a comment.\n");
   }
   return 0;
}
```

SAMPLE OUTPUT:

INPUT: //Hello world

OUTPUT: This is a single-line comment.

Output:

```
Enter a line: //hello world
This is a single-line comment.

--------------------------------
Process exited after 6.229 seconds with return value 0
Press any key to continue . . .
```

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 an operator: ");
   scanf("%c", &ch);
   if (ch == '+')
      printf("Valid Operator: Addition (+)\n");
   else if (ch == '-')
      printf("Valid Operator: Subtraction (-)\n");
   else if (ch == '*')
      printf("Valid Operator: Multiplication (*)\n");
   else if (ch == '/')
      printf("Valid Operator: Division (/)\n");
   else
      printf("Invalid Operator!\n");
   return 0;
}
```

SAMPLE OUTPUT:

INPUT: Enter an operator: *

OUTPUT: Valid Operator: Multiplication (*)

OUTPUT:

```
Enter an operator: *
Valid Operator: Multiplication (*)

--------------------------------
Process exited after 3.488 seconds with return value 0
Press any key to continue . . .
```

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, newlines = 0;
   printf("Enter text (press Ctrl+Z then Enter to stop):\n");
   while ((ch = getchar()) != EOF) {
      if (ch == ' ' || ch == '\t')
         spaces++;
      else if (ch == '\n')
         newlines++;
   }
   printf("\nNumber of whitespaces: %d\n", spaces);
   printf("Number of newline characters: %d\n", newlines);
   return 0;
}
```

SAMPLE OUTPUT:

Enter text (press Ctrl+Z then Enter to stop):

HELLO WORLD

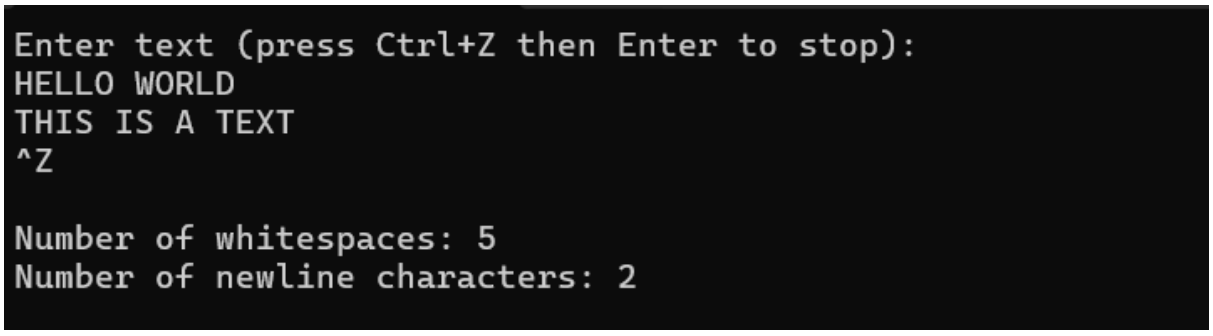
THIS IS A TEXT

^Z

Number of whitespaces: 4

Number of newline characters: 2

OUTPUT:



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

CODE:

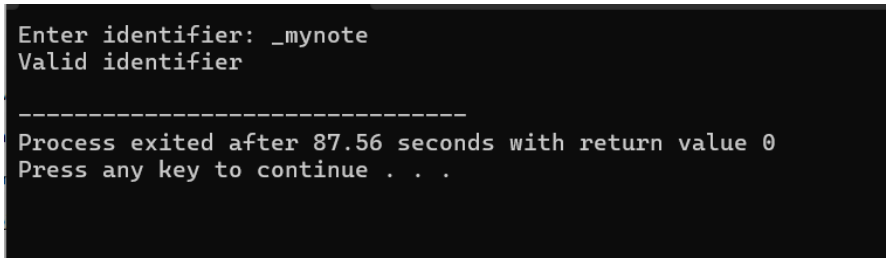
```
#include <stdio.h>
#include <ctype.h>
int main() {
   char id[100];
   int i;
   printf("Enter identifier: ");
   scanf("%s", id);
   if(!(isalpha(id[0]) || id[0]=='_')) {
      printf("Not a valid identifier\n");
      return 0;
   }
   for(i=1; id[i]!='\0'; i++) {
      if(!(isalnum(id[i]) || id[i]=='_')) {
         printf("Not a valid identifier\n");
         return 0;
      }
   }
   printf("Valid identifier\n");
   return 0;
}
```

SAMPLE OUTPUT:

INPUT: enter an operator:_mynote

OUTPUT: VALID IDENTIFIER

OUTPUT:


Enter identifier: _mynote
Valid identifier

Process exited after 87.56 seconds with return value 0
Press any key to continue . . .