UIT1601 - Compiler Design
Lex Programs

Name:	M Badri Narayanan
Reg No:	185002018
Semester:	VI
Date:	April 3, 2021

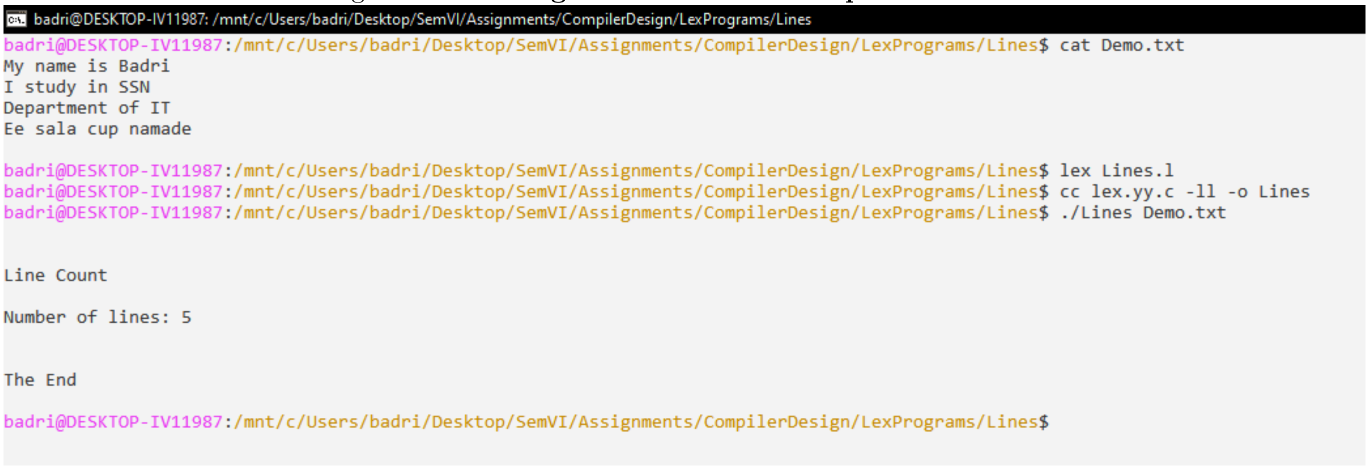
1 Counting number of lines

Program

```
%{
#include <stdio.h>
#include <string.h>
int lines = 0;
%}
%%
\n {lines++;}
. {;}
%%
int main(int argc, char **argv)
{
    if(argc != 2)
    {
        fprintf(stderr,"Please Enter file as second argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if(yyin == NULL)
    {
        fprintf(stderr,"File not found!\n");
        return 1;
    }
    yylex();
    printf("\n\nLine Count\n\n");
    printf("Number of lines: %d\n", lines);
    printf("\n\nThe End\n\n");
}
```

Output

Figure 1: **Counting number of lines Output**



```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ cat Demo.txt
My name is Badri
I study in SSN
Department of IT
Ee sala cup namade

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ lex Lines.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ cc lex.yy.c -ll -o Lines
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ ./Lines Demo.txt


Line Count

Number of lines: 5


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$
```

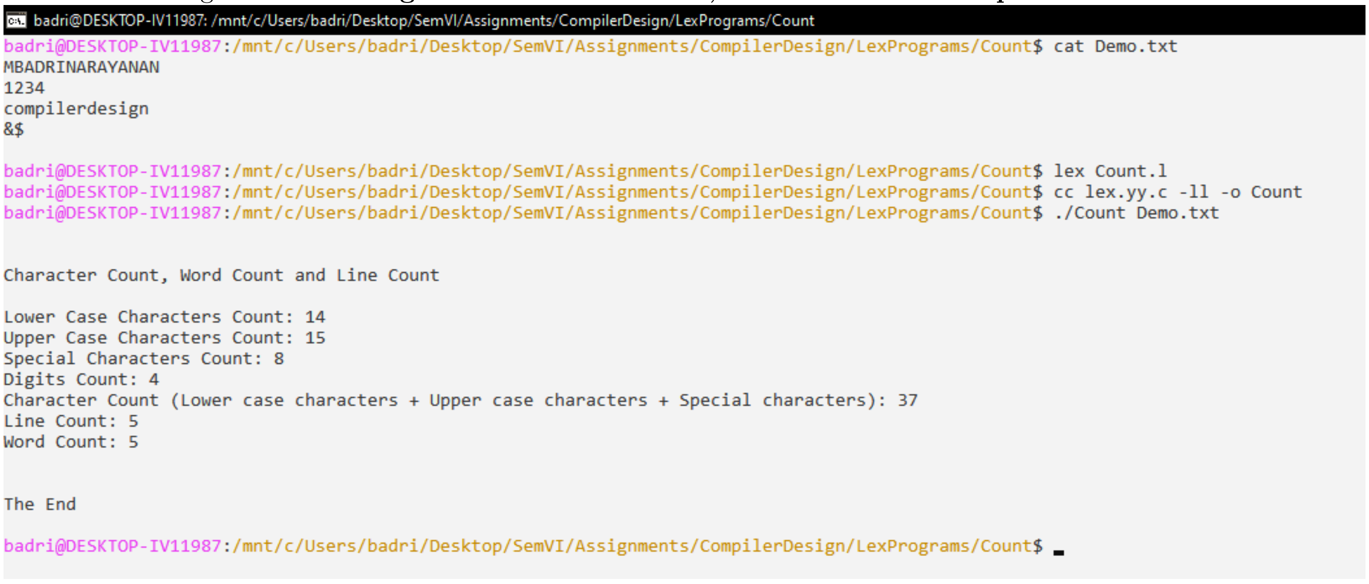
2 Counting number of characters, lines and words

Program

```
%{
        #include <stdio.h>
        int lines = 0;
        int words = 0;
        int lc = 0;
        int uc = 0;
        int digits = 0;
        int spl_char = 0;
        int total = 0;
%}
%%
\n { lines++; words++;}
[\t' '] words++;
[A-Z] uc++;
[a-z] lc++;
[0-9] digits++;
. spl_char++;
%%
int main(int argc, char **argv)
{
    if(argc != 2)
    {
        fprintf(stderr,"Please Enter file as second argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if(yyin == NULL)
    {
        fprintf(stderr,"File not found! \n");
        return 1;
    }
    yylex();
    total = lc + uc + spl_char;
    printf("\n\nCharacter Count, Word Count and Line Count \n\n");
    printf("Lower Case Characters Count: %d \n", lc);
    printf("Upper Case Characters Count: %d \n", uc);
    printf("Special Characters Count: %d \n", spl_char);
    printf("Digits Count: %d \n", digits);
    printf("Character Count (Lower case characters + Upper case characters + Special
    ↪  characters): %d \n", total);
    printf("Line Count: %d \n", lines);
    printf("Word Count: %d \n", words);
    printf("\n\nThe End \n\n");
}

int yywrap()
{
        return(1);
}
```

Output

Figure 2: **Counting number of characters, lines and words Output**



```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count$ cat Demo.txt
MBADRINARAYANAN
1234
compilerdesign
&$

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count$ lex Count.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count$ cc lex.yy.c -ll -o Count
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count$ ./Count Demo.txt


Character Count, Word Count and Line Count

Lower Case Characters Count: 14
Upper Case Characters Count: 15
Special Characters Count: 8
Digits Count: 4
Character Count (Lower case characters + Upper case characters + Special characters): 37
Line Count: 5
Word Count: 5


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Count$ _
```

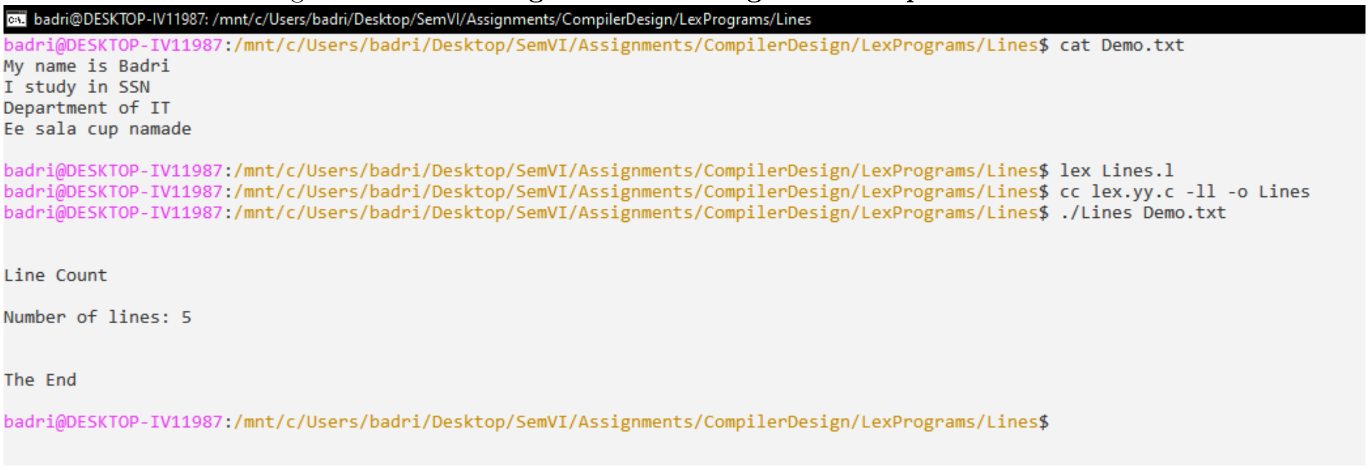
3 Count the number of numbers appearing in the input. Count the number of integers (without a decimal) separately from the number of floating-point numbers (with a decimal, and at least one digit on either side of the decimal).

Program

```
%{
#include <stdio.h>
#include <string.h>
int lines = 0;
%}
%%
\n {lines++;}
. {;}
%%
int main(int argc, char **argv)
{
    if(argc != 2)
    {
        fprintf(stderr,"Please Enter file as second argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if(yyin == NULL)
    {
        fprintf(stderr,"File not found!\n");
        return 1;
    }
    yylex();
    printf("\n\nLine Count\n\n");
    printf("Number of lines: %d\n", lines);
    printf("\n\nThe End\n\n");
}
```

Output

Figure 3: **Count Integers and Floating Point Output**



```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ cat Demo.txt
My name is Badri
I study in SSN
Department of IT
Ee sala cup namade

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ lex Lines.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ cc lex.yy.c -ll -o Lines
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$ ./Lines Demo.txt


Line Count

Number of lines: 5


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Lines$
```

4 Implement lexical analyzer using lex tool.

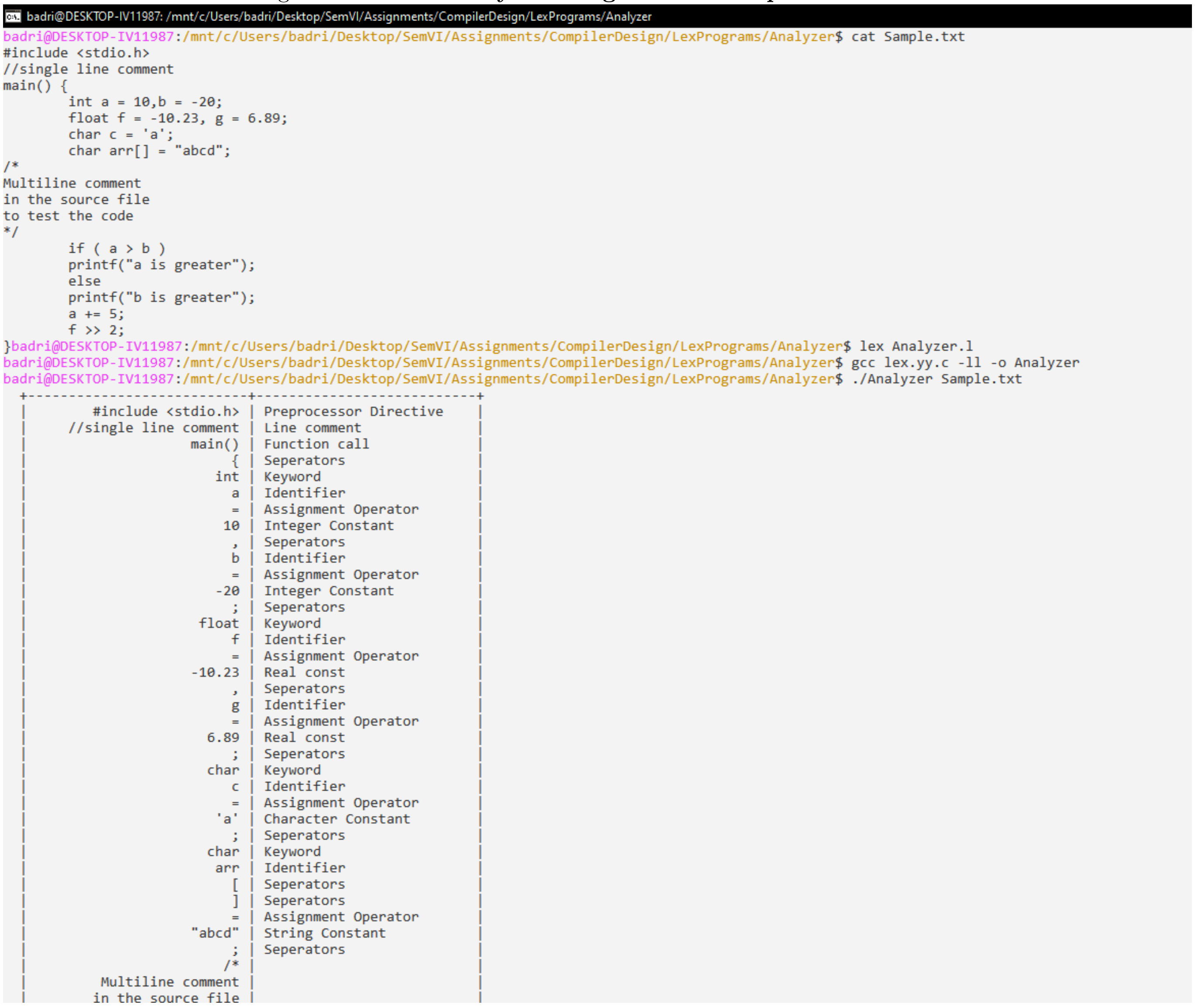
Program

```
%{
#include <stdio.h>
#include <string.h>
%}

pre_process ^#(.)*
line_comment    \/\/(.)*
multi_comment   \/\*(.|\n)*\*\/

keyword
↪   auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|lon

id          [a-zA-Z_]([a-zA-Z0-9_])*
function    {id}\((.)*\)

realConst (\+|\-)?[1-9][0-9]*\.[0-9]+
intConst (\+|\-)?[1-9][0-9]*
charConst   \'[a-zA-Z]\'
stringConst    \"[a-z A-Z]*\"

assignOp =
bitwiseOp "^"|"&"|"|"|"<<"|">>"
arithAssignOp "+="|"-="|"*="|"\="|"%="
relOp <|<=|>|>=|=|!=
arithOp "+"|"-"|"*"|"/"|"%"
logicOp &&|\|\||!
separators ";"|","|"."|"["|"]"|"("|")"|"{"|"}"|"["|"]"

/*printf("  | %25s | %-25s |\n", yytext, "Function call");*/
%%
{keyword}       {printf("  | %25s | %-25s |\n", yytext, "Keyword");}
{function}      {printf("  | %25s | %-25s |\n", yytext, "Function call");}
{id}            {printf("  | %25s | %-25s |\n", yytext, "Identifier");}
{realConst}     {printf("  | %25s | %-25s |\n", yytext, "Real const");}
{intConst}      {printf("  | %25s | %-25s |\n", yytext, "Integer Constant");}
{bitwiseOp}     {printf("  | %25s | %-25s |\n", yytext, "Bitwise Operator");}
{assignOp}      {printf("  | %25s | %-25s |\n", yytext, "Assignment Operator");}
{arithAssignOp} {printf("  | %25s | %-25s |\n", yytext, "Arith Assign Operator");}
{arithOp}       {printf("  | %25s | %-25s |\n", yytext, "Arithmetic Operator");}
{logicOp}       {printf("  | %25s | %-25s |\n", yytext, "Logical Operator");}
{relOp}         {printf("  | %25s | %-25s |\n", yytext, "Relational Operator");}
{charConst}     {printf("  | %25s | %-25s |\n", yytext, "Character Constant");}
{stringConst}   {printf("  | %25s | %-25s |\n", yytext, "String Constant");}
{separators}    {printf("  | %25s | %-25s |\n", yytext, "Seperators");}
{pre_process}   {printf("  | %25s | %-25s |\n", yytext, "Preprocessor Directive");}
{line_comment}  {printf("  | %25s | %-25s |\n", yytext, "Line comment");}
{multi_comment} {
    char *lines = strtok(yytext, "\n");
    while(lines){
        printf("  | %25s | ", lines);
        lines = strtok(NULL, "\n");
        printf("%-25s |\n",(lines!=NULL)?" ": "Multiline Comment");}
    }
.|\n|\r {}
```

```
%%
int main(int argc, char **argv)
{
    if(argc != 2)
    {
        fprintf(stderr,"Please Enter file as second argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if(yyin == NULL)
    {
        fprintf(stderr,"File not found!\n");
        return 1;
    }
    printf("  +---------------------------+---------------------------+\n");
    yylex();
    printf("  +---------------------------+---------------------------+\n");

}
```

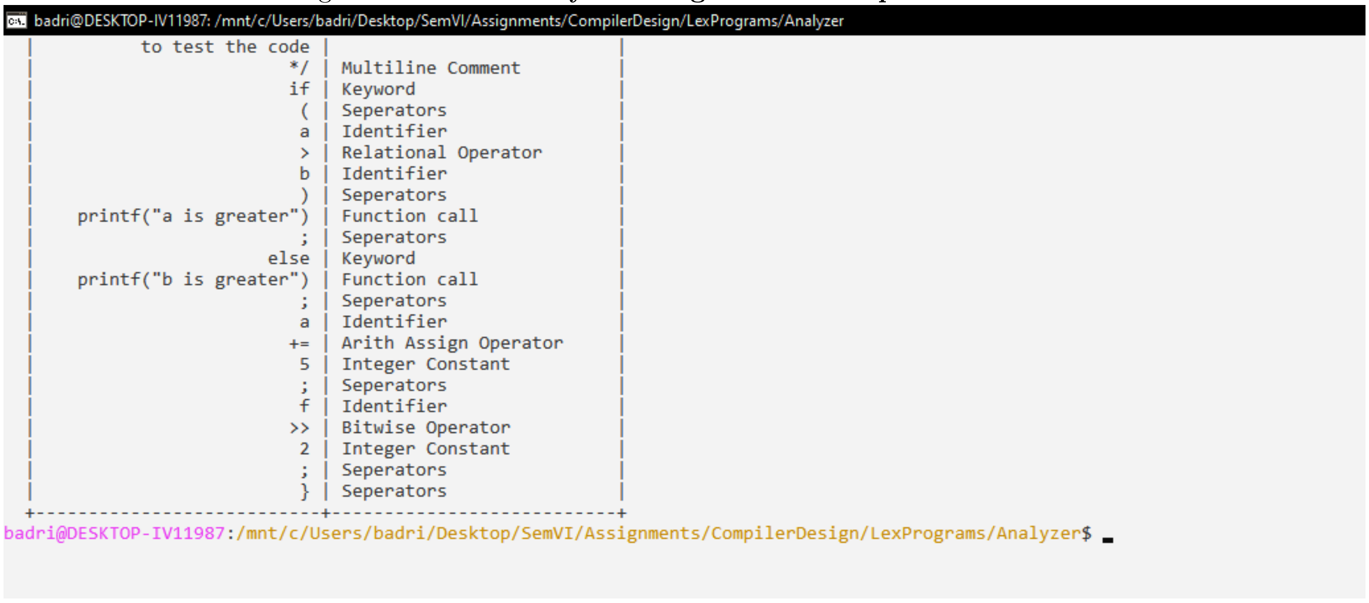
Output

Figure 4: **Lexical analyzer using lex tool Output**



```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer$ cat Sample.txt
#include <stdio.h>
//single line comment
main() {
        int a = 10,b = -20;
        float f = -10.23, g = 6.89;
        char c = 'a';
        char arr[] = "abcd";
/*
Multiline comment
in the source file
to test the code
*/
        if ( a > b )
        printf("a is greater");
        else
        printf("b is greater");
        a += 5;
        f >> 2;
}badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer$ lex Analyzer.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer$ gcc lex.yy.c -ll -o Analyzer
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer$ ./Analyzer Sample.txt
  +-------------------------+---------------------------+
  |       #include <stdio.h> | Preprocessor Directive    |
  |    //single line comment | Line comment              |
  |                   main() | Function call             |
  |                        { | Seperators                |
  |                      int | Keyword                   |
  |                        a | Identifier                |
  |                        = | Assignment Operator       |
  |                       10 | Integer Constant          |
  |                        , | Seperators                |
  |                        b | Identifier                |
  |                        = | Assignment Operator       |
  |                      -20 | Integer Constant          |
  |                        ; | Seperators                |
  |                    float | Keyword                   |
  |                        f | Identifier                |
  |                        = | Assignment Operator       |
  |                   -10.23 | Real const                |
  |                        , | Seperators                |
  |                        g | Identifier                |
  |                        = | Assignment Operator       |
  |                     6.89 | Real const                |
  |                        ; | Seperators                |
  |                     char | Keyword                   |
  |                        c | Identifier                |
  |                        = | Assignment Operator       |
  |                      'a' | Character Constant        |
  |                        ; | Seperators                |
  |                     char | Keyword                   |
  |                      arr | Identifier                |
  |                        [ | Seperators                |
  |                        ] | Seperators                |
  |                        = | Assignment Operator       |
  |                   "abcd" | String Constant           |
  |                        ; | Seperators                |
  |                       /* |                           |
  |        Multiline comment |                           |
  |       in the source file |                           |
```

Figure 5: **Lexical analyzer using lex tool Output**



```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer
 |          to test the code |                            |
 |                        */ | Multiline Comment          |
 |                        if | Keyword                    |
 |                         ( | Seperators                 |
 |                         a | Identifier                 |
 |                         > | Relational Operator        |
 |                         b | Identifier                 |
 |                         ) | Seperators                 |
 |    printf("a is greater") | Function call              |
 |                         ; | Seperators                 |
 |                      else | Keyword                    |
 |    printf("b is greater") | Function call              |
 |                         ; | Seperators                 |
 |                         a | Identifier                 |
 |                        += | Arith Assign Operator      |
 |                         5 | Integer Constant           |
 |                         ; | Seperators                 |
 |                         f | Identifier                 |
 |                        >> | Bitwise Operator           |
 |                         2 | Integer Constant           |
 |                         ; | Seperators                 |
 |                         } | Seperators                 |
 +---------------------------+----------------------------+
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/LexPrograms/Analyzer$
```