SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1602 - Compiler Design

EX - 2 : Implementation of Lexical Analyzer using LEX tool

NAME : Gayathri M

REG NO: 185001050

DATE : 16/02/2021

Program Code:

```
%{
#include <stdio.h>
%}
anyChar .|\n
multiline \/\*{anyChar}*\*\/
single \/\/.*
dir #.*
str \".*\"
c \'.\'
digit [0-9]
dec digit+\.digit+
hex [0-9A-Fa-f]+[hH]
arith [+\-*/%]
logical (&&)|(\|\|)|(!)
relational \<|\<=|\>|\>=|==|!=
bitwise (\^)|&|(\|)|(\<\<)|(\>\>)
unary \+\+|\-\-
special [;|,|\.|\[|\]|\{|\}|\(|\)]

%%
{multiline} {printf("%-30s \t\t- Multiline comment\n",yytext);}
```

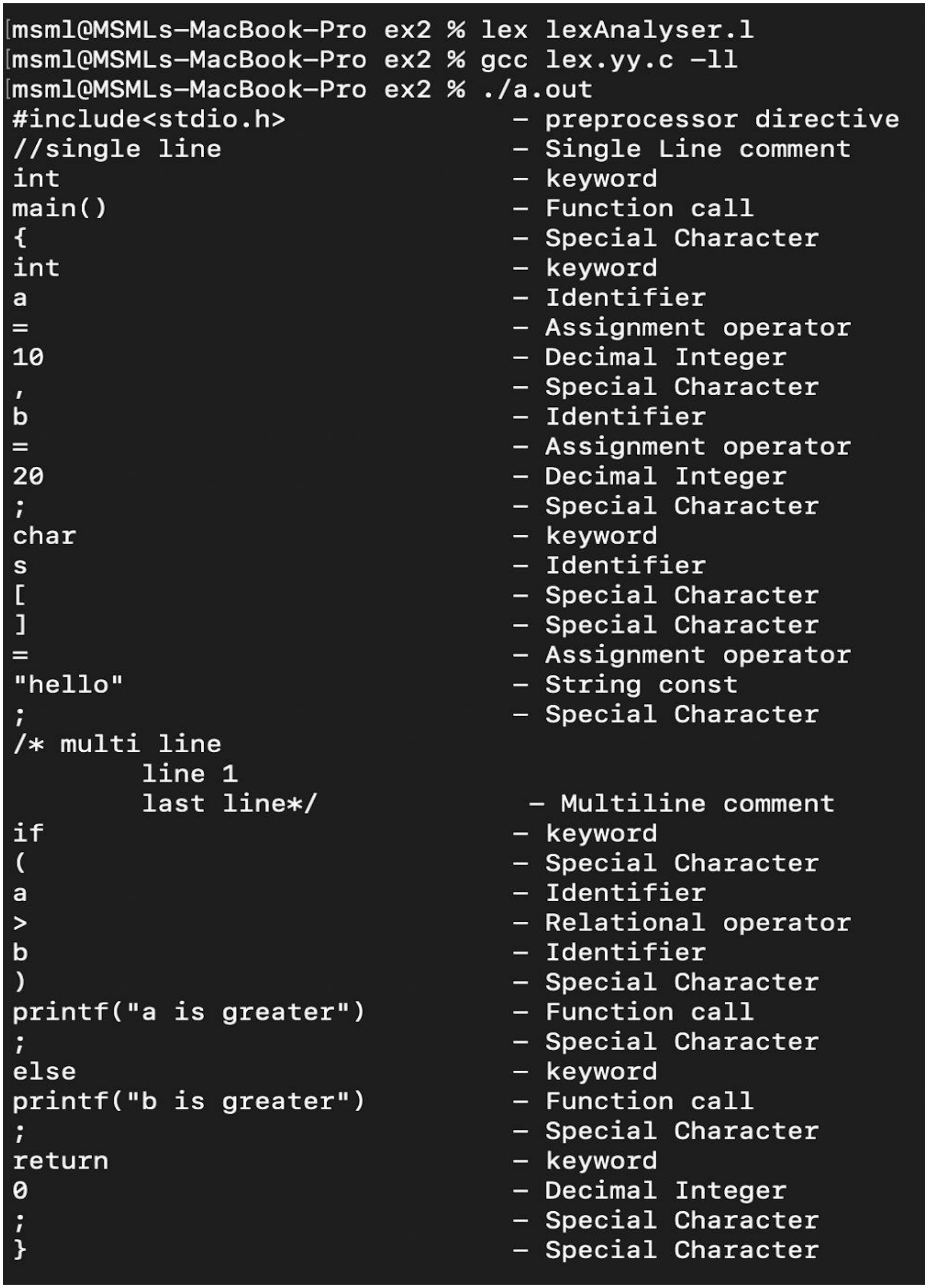
```
{single} { printf("%-30s - Single Line comment\n", yytext); }

{dir} { printf("%-30s - preprocessor directive\n",yytext);}

auto|break|case|char|const|continue|default|do|double|else|enum|extern
|float |
for|goto|if|int|long|register|return|short|signed|sizeof |
static|struct|switch|typedef|union|unsigned|void|volatile |
while {printf("%-30s - keyword\n",yytext);}

{str} { printf("%-30s - String const\n", yytext); }
{c} { printf("%-30s - Character const\n", yytext); }

{dec}  { printf("%-30s - Double\n", yytext); }
{hex} { printf("%-30s - Hexadecimal Integer\n", yytext); }

{digit}+ { printf("%-30s - Decimal Integer\n", yytext); }

{arith}= { printf("%-30s - Arithmetic assignment operators\n",
yytext); }
{arith} { printf("%-30s - Arithmetic operators\n", yytext); }

{logical} { printf("%-30s - Logical operator\n", yytext); }
{relational} { printf("%-30s - Relational operator\n", yytext); }

{bitwise} { printf("%-30s - Bitwise operator\n", yytext); }
{unary} { printf("%-30s - Unary operator\n", yytext); }

= { printf("%-30s - Assignment operator\n", yytext); }
{special} { printf("%-30s - Special Character\n", yytext); }

[a-zA-Z_][a-zA-Z0-9_]*\(.*\) { printf("%-30s - Function call\n",
yytext); }
[a-zA-Z_][a-zA-Z0-9_]* { printf("%-30s - Identifier\n", yytext); }

.|\n { }
%%
int main()
{
```

```
FILE *file;
file=fopen("source.c","r");
if(!file)
{
printf("could not open the file");
exit(0);
}
yyin=file;
yylex();
printf("\n");
return(0);
}
```

<u>SOURCE CODE</u> <u>(source.txt) :</u>

```
#include<stdio.h>
//single line
int main(){
     int a=10,b=20;
     char s[] = "hello";
     /* multi line
     line 1
     last line*/
     if (a>b)
          printf("a is greater");
     else
          printf("b is greater");
     return 0;
}
```

SAMPLE OUTPUT :



```
[msml@MSMLs-MacBook-Pro ex2 % lex lexAnalyser.l
[msml@MSMLs-MacBook-Pro ex2 % gcc lex.yy.c -ll
[msml@MSMLs-MacBook-Pro ex2 % ./a.out
#include<stdio.h>                 - preprocessor directive
//single line                     - Single Line comment
int                               - keyword
main()                            - Function call
{                                 - Special Character
int                               - keyword
a                                 - Identifier
=                                 - Assignment operator
10                                - Decimal Integer
,                                 - Special Character
b                                 - Identifier
=                                 - Assignment operator
20                                - Decimal Integer
;                                 - Special Character
char                              - keyword
s                                 - Identifier
[                                 - Special Character
]                                 - Special Character
=                                 - Assignment operator
"hello"                           - String const
;                                 - Special Character
/* multi line
        line 1
        last line*/                - Multiline comment
if                                - keyword
(                                 - Special Character
a                                 - Identifier
>                                 - Relational operator
b                                 - Identifier
)                                 - Special Character
printf("a is greater")            - Function call
;                                 - Special Character
else                              - keyword
printf("b is greater")            - Function call
;                                 - Special Character
return                            - keyword
0                                 - Decimal Integer
;                                 - Special Character
}                                 - Special Character
```

<u>Learning Outcomes :</u>

- I understood the use and necessity of lexical analyser in a compiler.
- I learnt about regular expressions.
- I learnt how to use regular expressions in LEX.
- I learnt to design a basic lexical analyser using the LEX tool.