****

Name: Anish Vaidya
Roll: 62
Div: A
SRN: 201900160

****

Compiler Design Assignment-2: Design a Lexical analyzer for the subset of C Language using LEX or FLEX.

Input Code:

```
void main()
{
	int a=10;
	int b=20;
	char c="charvalue";
	int 62anish;
	printf("anish10");
}
```

OUTPUT:

```
Line | Lexeme |       Token

 1     void          Keyword
 1     main          Identifier
 1     (             Delimiter
 1     )             Delimiter
 2     {             Delimiter
 3     int           Keyword
```

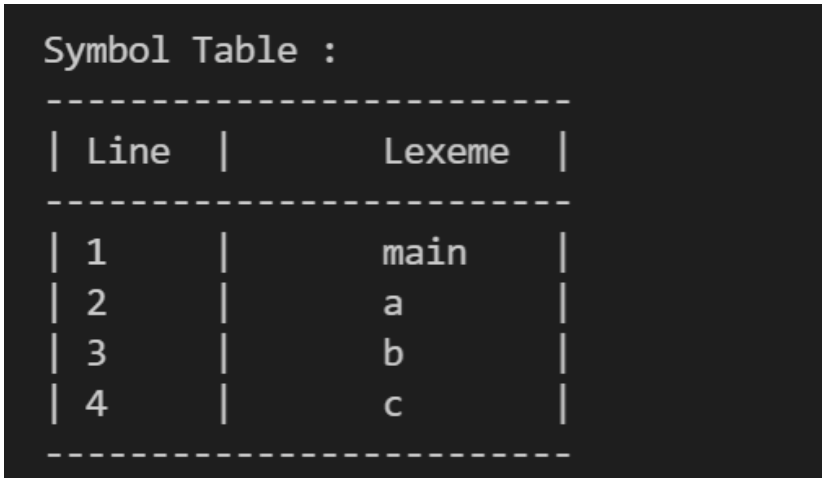
3	a	Identifier
3	=	Assignment Operator
3	10	Constant
3	;	Delimiter
4	int	Keyword
4	b	Identifier
4	=	Assignment Operator
4	20	Constant
4	;	Delimiter
5	char	Keyword
5	c	Identifier
5	=	Assignment Operator
5	"charvalue"	String Constant/Literal
5	;	Delimiter
6	int	Keyword
6	62anish	ERROR This is ILLEGAL
6	;	Delimiter
7	printf	Keyword
7	(	Delimiter
7	"anish10"	String Constant/Literal
7	)	Delimiter
7	;	Delimiter
8	}	Delimiter

```
PS D:\COLLEGEWORK\SEMESTER-6\Compiler Design\Assigment2_Final> flex sectry.l
PS D:\COLLEGEWORK\SEMESTER-6\Compiler Design\Assigment2_Final> gcc lex.yy.c
PS D:\COLLEGEWORK\SEMESTER-6\Compiler Design\Assigment2_Final> ./a.exe ./input.txt

Line | Lexeme |         Token

 1      void            Keyword
 1      main            Identifier
 1      (               Delimiter
 1      )               Delimiter
 2      {               Delimiter
 3      int             Keyword
 3      a               Identifier
 3      =               Assignment Operator
 3      10              Constant
 3      ;               Delimiter
 4      int             Keyword
 4      b               Identifier
 4      =               Assignment Operator
 4      20              Constant
```

```
 4      20              Constant
 4      ;               Delimiter
 5      char            Keyword
 5      c               Identifier
 5      =               Assignment Operator
 5      "charvalue"     String Constant/Literal
 5      ;               Delimiter
 6      int             Keyword
 6      62anish         ERROR This is ILLEGAL
 6      ;               Delimiter
 7      printf          Keyword
 7      (               Delimiter
 7      "anish10"       String Constant/Literal
 7      )               Delimiter
 7      ;               Delimiter
 8      }               Delimiter
```



Source Code:

```
%{
#include <stdio.h>
#include<string.h>
struct symEntry{
  int index;
  char lexeme[30];
};

struct symEntry symtable[30];
int sti=0;
int line=1;


void put_symtab(){
  int j;
  if(sti==0){
    symtable[sti].index=sti+1;
    strcpy(symtable[sti].lexeme,yytext);
    sti++;
    return;
```

```
  }
  for(j=0;j<sti;j++){
    if(strcmp(symtable[j].lexeme,yytext)==0){
      return;
    }
  }
  symtable[sti].index=sti+1;
  strcpy(symtable[sti].lexeme,yytext);
  sti++;
}

%}

letter [a-zA-Z]
number [0-9]
delim ["|']

%%
"int"|"if"|"double"|"long"|"goto"|"static"|"float"|"short"|"while"|"char"|"const"|"voi
d"|"else"|"return"|"printf"|"scanf"       {printf("\n %d\t%s \t\tKeyword",
line,yytext);}

"("|")"|"{"|"}"|"["|"]"|";"|","              {printf("\n %d\t%s \t\tDelimiter", line, yytext);}

{delim}               {printf("\n %d\t%s \t\tDelimiter", line, yytext);}

"+"|"-"|"*"|"%"|"/"|"++"|"--"          {printf("\n %d\t%s \t\tArithmetic
Operator",line, yytext);}
"=="|"<"|">"|"<="|">="       {printf("\n %d\t%s \t\tRelational Operator",line,
yytext);}
"="                                          {printf("\n %d\t%s \t\tAssignment
Operator",line, yytext);}
```

```
{letter}+|({letter}{number})*          {printf("\n %d\t%s \t\tIdentifier",line,
yytext); put_symtab();}
{number}+                                                  {printf("\n %d\t%s
\t\tConstant",line, yytext);}

{number}+{letter}+              {printf("\n %d\t%s \tERROR This is
ILLEGAL",line,yytext);}


{delim}({letter}|{number})*{delim}          {printf("\n %d\t%s \tString
Constant/Literal",line,yytext);}


"\n"    {line++;}

%%


void print_st(){
  int j;
  printf("\nSymbol Table : \n");
  printf("-----------------------\n");
  printf("| Line\t|\tLexeme\t|\n");
  printf("-----------------------\n");
  for(j=0;j<sti;j++){
    printf("| %d\t|\t%s\t|\n",symtable[j].index,symtable[j].lexeme);
  }
  printf("-----------------------\n");
}
```

```
int main(int argc, char* argv[])
{

        yyin = fopen(argv[1], "r");
        printf("\nLine | Lexeme | \tToken\n");
        yylex();
   printf("\n\n");
   print_st();
        fclose(yyin);
}

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