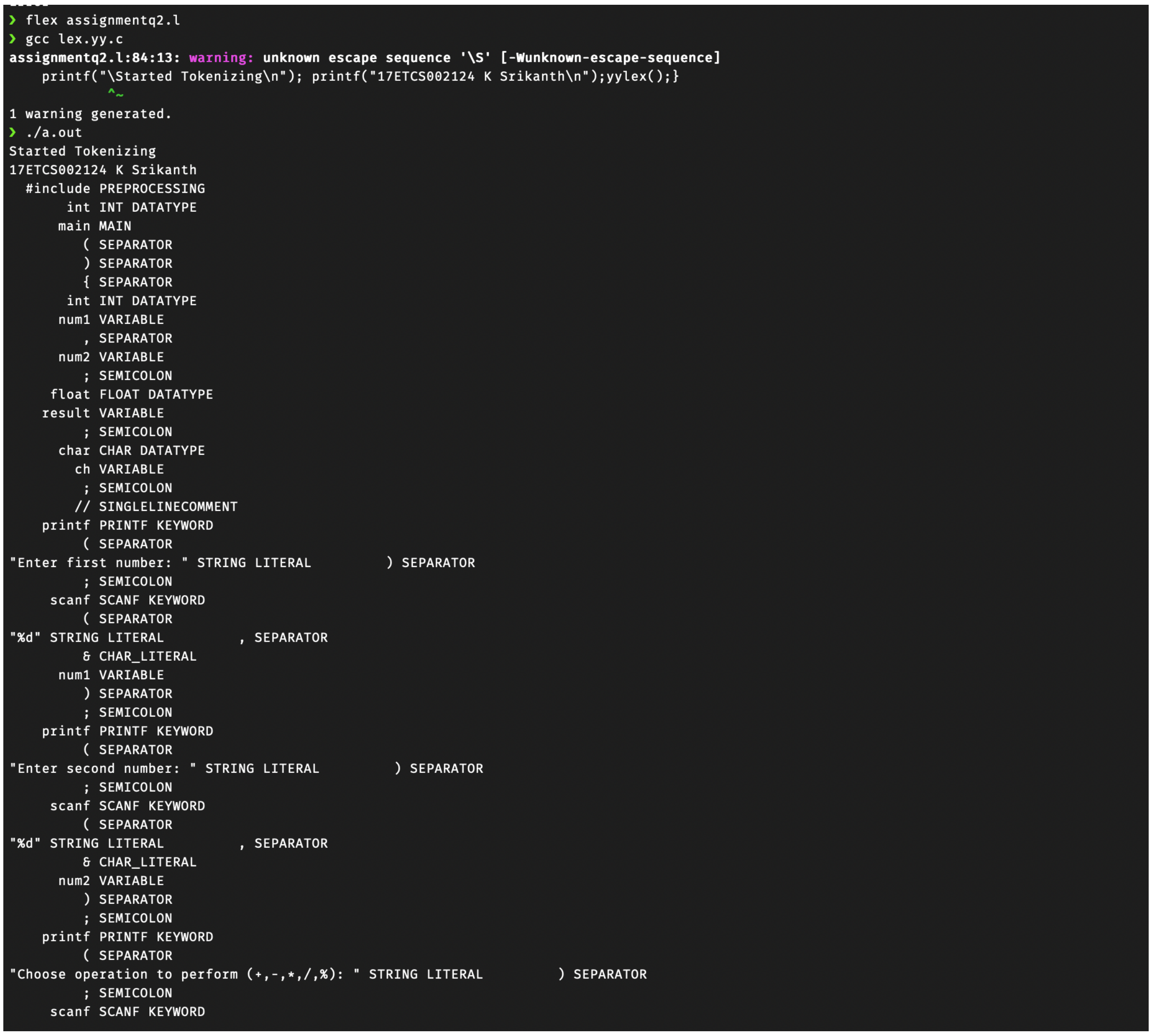
Name : K Srikanth   Registration Number : 17ETCS0012124

Lex Code

```
%{
    int yylex();
    void yyerror(char *s);
%}
alphabet [a-zA-Z] // substitute definitions
digit [0-9]
underscore \_
whitespace [ \t\r\f\v]+
%x PREPROCESSING
%x MULTILINECOMMENT
%x SINGLELINECOMMENT
%%
<<EOF>> {exit(0);}
^"#include" {BEGIN PREPROCESSING; printf("%10s PREPROCESSING\n",yytext); }
<PREPROCESSING>{whitespace} ;
<PREPROCESSING>"<"[^<>\n]*">" {BEGIN INITIAL;}
<PREPROCESSING>\"[^<>\n]*\" {BEGIN INITIAL;}
<PREPROCESSING>"\n" {yylineno++; BEGIN INITIAL;}
<PREPROCESSING>. {yyerror("Mistake in Header");}
"/*" {BEGIN MULTILINECOMMENT; printf("%10s MULTILINECOMMENT\n",yytext); }
<MULTILINECOMMENT>.|{whitespace} ;
<MULTILINECOMMENT>\n {yylineno++;}
<MULTILINECOMMENT>"*/" {BEGIN INITIAL;}
<MULTILINECOMMENT>"/*" {yyerror("Comment format invalid");}
"//" {BEGIN SINGLELINECOMMENT; printf("%10s SINGLELINECOMMENT\n",yytext); }
<SINGLELINECOMMENT>\n {yylineno++; BEGIN INITIAL;}
<SINGLELINECOMMENT>. ;
\+ {printf("%10s PLUS\n",yytext);}
\- {printf("%10s MINUS\n",yytext);}
\* {printf("%10s MULT\n",yytext);}
\/ {printf("%10s DIV\n",yytext);}
\^ {printf("%10s POW\n",yytext);}
"%" {printf("%10s MOD ARITHMETIC OPERATOR\n",yytext);}
"--" {printf("%10s DECREMENT ARITHMETIC OPERATOR\n",yytext);}
"++" {printf("%10s INCREMENT ARITHMETIC OPERATOR\n",yytext);}
">" {printf("%10s GT COMPARISION OPERATOR\n",yytext);}
"<" {printf("%10s LT COMPARISION OPERATOR\n",yytext);}
">=" {printf("%10s GT_EQ COMPARISION OPERATOR\n",yytext);}
"<=" {printf("%10s LT_EQ COMPARISION OPERATOR\n",yytext);}
"==" {printf("%10s EQUAL COMPARISION OPERATOR\n",yytext);}
"!=" {printf("%10s NOT_EQUAL COMPARISION OPERATOR\n",yytext);}
"||" {printf("%10s OR LOGICAL OPERATOR\n",yytext);}
"&&" {printf("%10s AND LOGICAL OPERATOR\n",yytext);}
"!" {printf("%10s NOT LOGICAL OPERATOR\n",yytext);}
main {printf("%10s MAIN\n",yytext);}
printf {printf("%10s PRINTF KEYWORD\n",yytext);}
scanf {printf("%10s SCANF KEYWORD\n",yytext);}
return {printf("%10s RETURN TYPE\n",yytext);}
switch {printf("%10s SWITCH STATEMENT\n",yytext);}
case {printf("%10s CASE STATEMENT\n",yytext);}
default {printf("%10s DEFAULT STATEMENT\n",yytext);}
break {printf("%10s BREAK STATEMENT\n",yytext);}
int {printf("%10s INT DATATYPE\n",yytext);}
float {printf("%10s FLOAT DATATYPE\n",yytext);}
char {printf("%10s CHAR DATATYPE\n",yytext);}
";" {printf("%10s SEMICOLON\n",yytext);}
[\(\)\{\}\,\[\]] {printf("%10s SEPARATOR\n",yytext);}
\".*\" {printf("%s STRING LITERAL",yytext);}
([_a-zA-Z]+[_a-zA-Z0-9]*) {printf("%10s VARIABLE\n",yytext);}
{digit}+ { printf("%10s INT VALUE\n",yytext);}
{digit}+[\.]{digit}+ { printf("%10s FLOAT VALUE\n",yytext);}
\n ;
{whitespace} ;
. {printf("%10s CHAR_LITERAL\n",yytext);}
%%
int yywrap(){ return 1;}
void yyerror (char *s) {fprintf (stderr, "%s at line %d\n", s, yylineno);}
int main() {
    yyin = fopen("input.c", "r");
    if(yyin==NULL) printf("\nError\n");
    else{
    printf("\Started Tokenizing\n"); printf("17ETCS002124 K Srikanth\n");yylex();}
    fclose(yyin);
    return 0; }
```

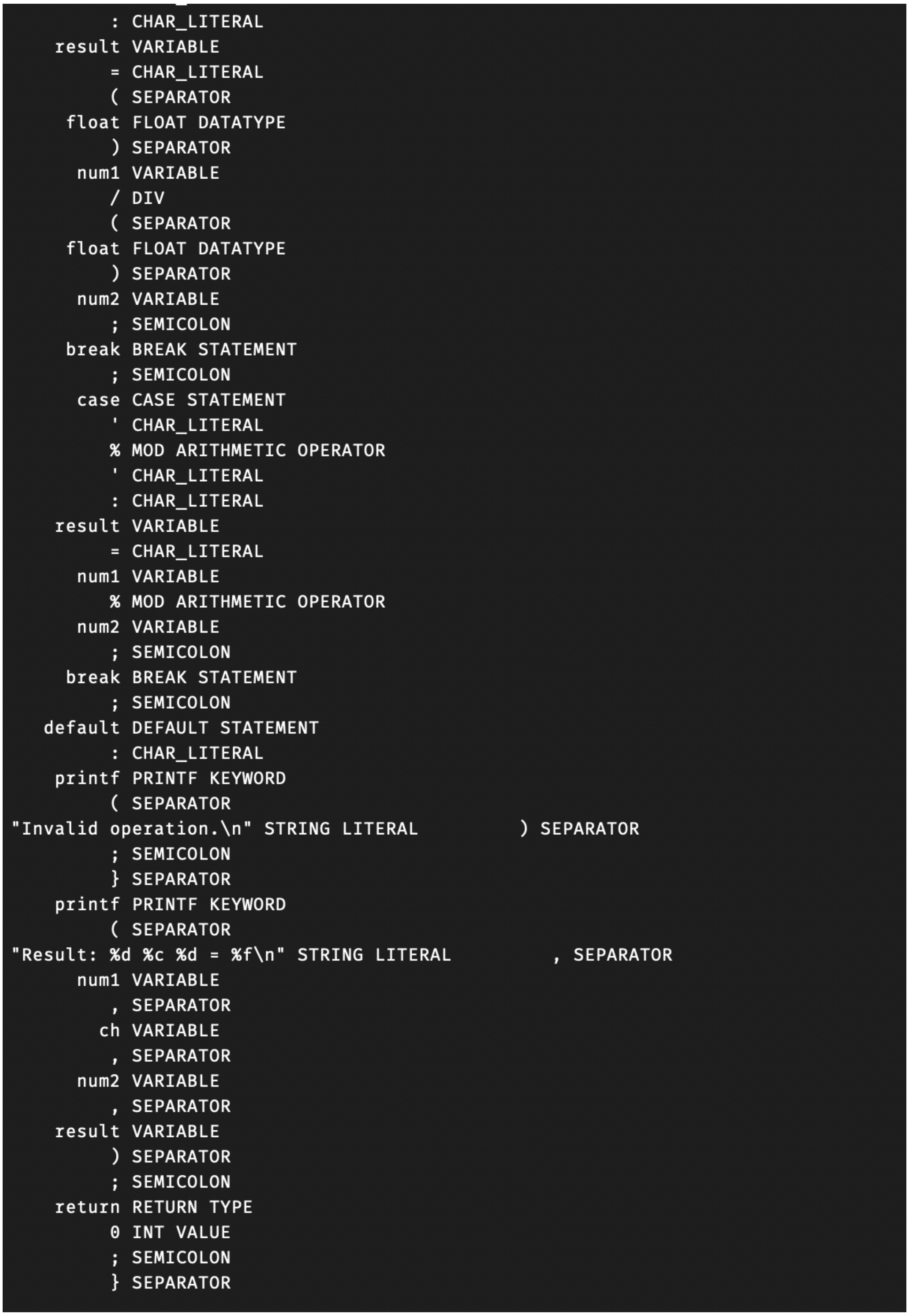
Name : K Srikanth   Registration Number : 17ETCS0012124

Lex Output



```
❯ flex assignmentq2.l
❯ gcc lex.yy.c
assignmentq2.l:84:13: warning: unknown escape sequence '\S' [-Wunknown-escape-sequence]
    printf("\Started Tokenizing\n"); printf("17ETCS002124 K Srikanth\n");yylex();}
            ^~
1 warning generated.
❯ ./a.out
Started Tokenizing
17ETCS002124 K Srikanth
  #include PREPROCESSING
       int INT DATATYPE
      main MAIN
         ( SEPARATOR
         ) SEPARATOR
         { SEPARATOR
       int INT DATATYPE
      num1 VARIABLE
         , SEPARATOR
      num2 VARIABLE
         ; SEMICOLON
     float FLOAT DATATYPE
    result VARIABLE
         ; SEMICOLON
      char CHAR DATATYPE
        ch VARIABLE
         ; SEMICOLON
        // SINGLELINECOMMENT
    printf PRINTF KEYWORD
         ( SEPARATOR
"Enter first number: " STRING LITERAL         ) SEPARATOR
         ; SEMICOLON
     scanf SCANF KEYWORD
         ( SEPARATOR
"%d" STRING LITERAL         , SEPARATOR
         & CHAR_LITERAL
      num1 VARIABLE
         ) SEPARATOR
         ; SEMICOLON
    printf PRINTF KEYWORD
         ( SEPARATOR
"Enter second number: " STRING LITERAL         ) SEPARATOR
         ; SEMICOLON
     scanf SCANF KEYWORD
         ( SEPARATOR
"%d" STRING LITERAL         , SEPARATOR
         & CHAR_LITERAL
      num2 VARIABLE
         ) SEPARATOR
         ; SEMICOLON
    printf PRINTF KEYWORD
         ( SEPARATOR
"Choose operation to perform (+,-,*,/,%): " STRING LITERAL         ) SEPARATOR
         ; SEMICOLON
     scanf SCANF KEYWORD
```

Figure 1 Lex Token generation for input C



```
         : CHAR_LITERAL
    result VARIABLE
         = CHAR_LITERAL
         ( SEPARATOR
     float FLOAT DATATYPE
         ) SEPARATOR
      num1 VARIABLE
         / DIV
         ( SEPARATOR
     float FLOAT DATATYPE
         ) SEPARATOR
      num2 VARIABLE
         ; SEMICOLON
     break BREAK STATEMENT
         ; SEMICOLON
      case CASE STATEMENT
         ' CHAR_LITERAL
         % MOD ARITHMETIC OPERATOR
         ' CHAR_LITERAL
         : CHAR_LITERAL
    result VARIABLE
         = CHAR_LITERAL
      num1 VARIABLE
         % MOD ARITHMETIC OPERATOR
      num2 VARIABLE
         ; SEMICOLON
     break BREAK STATEMENT
         ; SEMICOLON
   default DEFAULT STATEMENT
         : CHAR_LITERAL
    printf PRINTF KEYWORD
         ( SEPARATOR
"Invalid operation.\n" STRING LITERAL         ) SEPARATOR
         ; SEMICOLON
         } SEPARATOR
    printf PRINTF KEYWORD
         ( SEPARATOR
"Result: %d %c %d = %f\n" STRING LITERAL         , SEPARATOR
      num1 VARIABLE
         , SEPARATOR
        ch VARIABLE
         , SEPARATOR
      num2 VARIABLE
         , SEPARATOR
    result VARIABLE
         ) SEPARATOR
         ; SEMICOLON
    return RETURN TYPE
         0 INT VALUE
         ; SEMICOLON
         } SEPARATOR
```

Figure 2 Lex Token generation for input C Continued