Compiler Design Lab

CLASS WORK - 1

31.01.2022

—

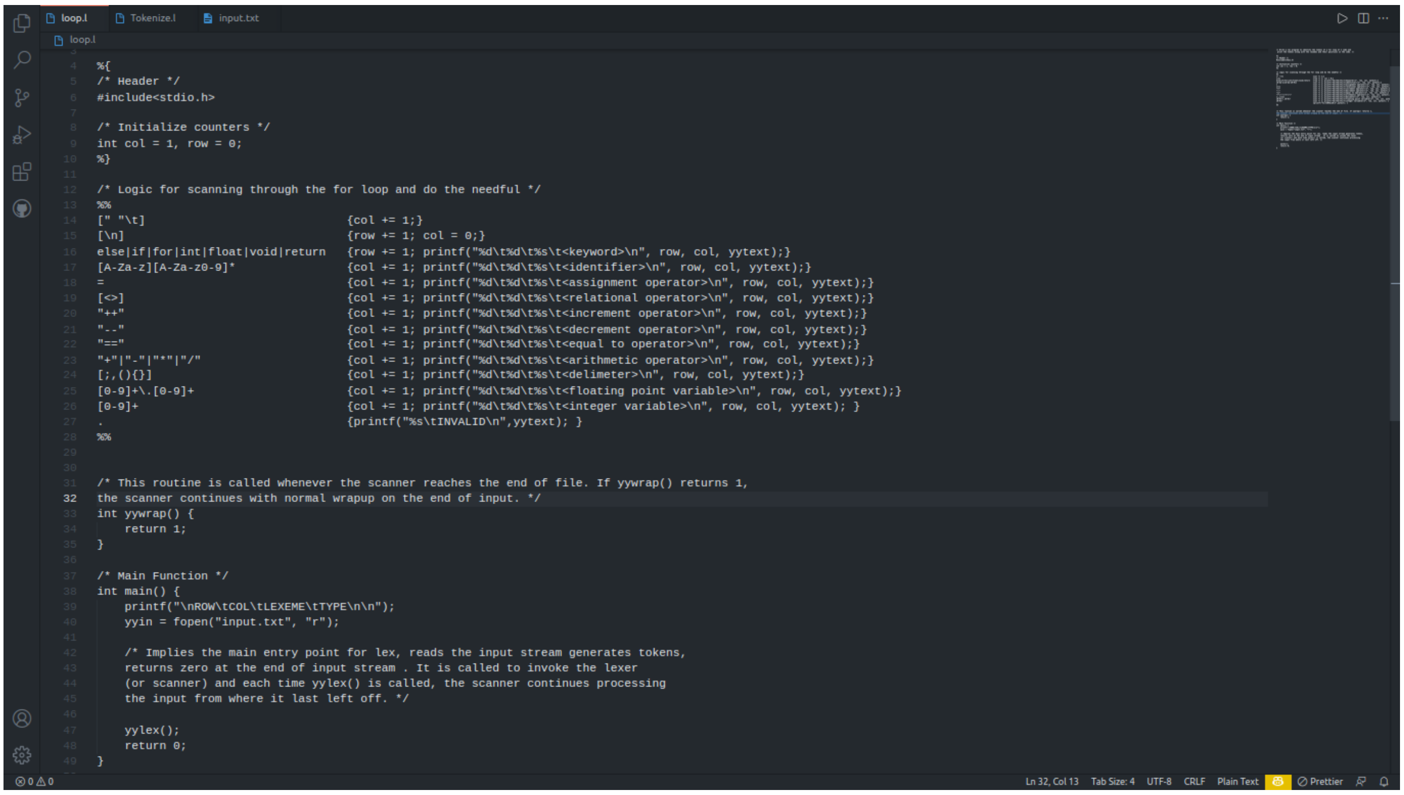
Atanu Ghosh
Roll: **001910501005**
Section: A-1
BCSE-III (2019-2023) 6th sem

The following code has been implemented and tested with GNU Flex 2.6.4 and GCC 9.3.0 on Linux Operating System (Distro Choice: Ubuntu 20.04 LTS).

Question

Write a lex program to identify the tokens of a for loop of C code and print the tokens along with the lexemes and their positions in the code.

Code Snippet



```
%{
/* Header */
#include<stdio.h>

/* Initialize counters */
int col = 1, row = 0;
%}

/* Logic for scanning through the for loop and do the needful */
%%
[" "\t]                             {col += 1;}
[\n]                                {row += 1; col = 0;}
else|if|for|int|float|void|return   {row += 1; printf("%d\t%d\t%s\t<keyword>\n", row, col, yytext);}
[A-Za-z][A-Za-z0-9]*                {col += 1; printf("%d\t%d\t%s\t<identifier>\n", row, col, yytext);}
=                                   {col += 1; printf("%d\t%d\t%s\t<assignment operator>\n", row, col, yytext);}
[<>]                                {col += 1; printf("%d\t%d\t%s\t<relational operator>\n", row, col, yytext);}
"++"                                {col += 1; printf("%d\t%d\t%s\t<increment operator>\n", row, col, yytext);}
"--"                                {col += 1; printf("%d\t%d\t%s\t<decrement operator>\n", row, col, yytext);}
"=="                                {col += 1; printf("%d\t%d\t%s\t<equal to operator>\n", row, col, yytext);}
"+"|"-"|"*"|"/"                     {col += 1; printf("%d\t%d\t%s\t<arithmetic operator>\n", row, col, yytext);}
[;,(){}]                            {col += 1; printf("%d\t%d\t%s\t<delimeter>\n", row, col, yytext);}
[0-9]+\.[0-9]+                      {col += 1; printf("%d\t%d\t%s\t<floating point variable>\n", row, col, yytext);}
[0-9]+                              {col += 1; printf("%d\t%d\t%s\t<integer variable>\n", row, col, yytext); }
.                                   {printf("%s\tINVALID\n",yytext); }
%%


/* This routine is called whenever the scanner reaches the end of file. If yywrap() returns 1,
the scanner continues with normal wrapup on the end of input. */
int yywrap() {
    return 1;
}

/* Main Function */
int main() {
    printf("\nROW\tCOL\tLEXEME\tTYPE\n\n");
    yyin = fopen("input.txt", "r");

    /* Implies the main entry point for lex, reads the input stream generates tokens,
    returns zero at the end of input stream . It is called to invoke the lexer
    (or scanner) and each time yylex() is called, the scanner continues processing
    the input from where it last left off. */

    yylex();
    return 0;
}
```

```
%{
/* Header */
#include<stdio.h>

/* Initialize counters */
int col = 1, row = 0;
%}

/* Logic for scanning through the for loop and do the needful */
%%
[" "\t]            {col += 1;}
[\n]               {row += 1; col = 0;}

else|if|for|int|float|void|return{
        row += 1; printf("%d\t%d\t%s\t<keyword>\n", row, col, yytext);
}

[A-Za-z][A-Za-z0-9]*   {
        col += 1; printf("%d\t%d\t%s\t<identifier>\n", row, col, yytext);
}

=       {col += 1; printf("%d\t%d\t%s\t<assignment operator>\n", row, col, yytext);}

[<>]    {col += 1; printf("%d\t%d\t%s\t<relational operator>\n", row, col, yytext);}

"++"    {col += 1; printf("%d\t%d\t%s\t<increment operator>\n", row, col, yytext);}

"--"    {col += 1; printf("%d\t%d\t%s\t<decrement operator>\n", row, col, yytext);}

"=="    {col += 1; printf("%d\t%d\t%s\t<equal to operator>\n", row, col, yytext);}

"+"|"-"|"*"|"/"    {col += 1; printf("%d\t%d\t%s\t<arithmetic operator>\n", row, col, yytext);}

[;,(){}]           {col += 1; printf("%d\t%d\t%s\t<delimeter>\n", row, col, yytext);}
[0-9]+\.[0-9]+      {col += 1; printf("%d\t%d\t%s\t<floating point variable>\n", row, col, yytext);}
[0-9]+             {col += 1; printf("%d\t%d\t%s\t<integer variable>\n", row, col, yytext); }
.       printf("%s\tINVALID\n",yytext); }

%%
```

```
/* This routine is called whenever the scanner reaches
the end of the file. If yywrap() returns 1, the scanner
continues with a normal wrapup on the end of input. */

int yywrap() {
        return 1;
}


/* Main Function */

int main() {

        printf("\nROW\tCOL\tLEXEME\tTYPE\n\n");

        /* Takes input from the file named as input.txt */
        yyin = fopen("input.txt", "r");


/* Implies the main entry point for lexing, reads the input stream generates
tokens, return zero at the end of the input stream. It is called to invoke the
Lexer (or scanner) and each time yylex() is called, the scanner continues
processing the input from where it last left off. */

        yylex();

        return 0;
}
```

P.S.- IF YOU GET WHILE COPY-PASTING AND RUNNING THE CODE, THE SOURCE FILE CAN BE FOUND HERE : https://drive.google.com/drive/folders/1fgkdP3hAOyPQLe3DOx_i8j_vMKcogbbE

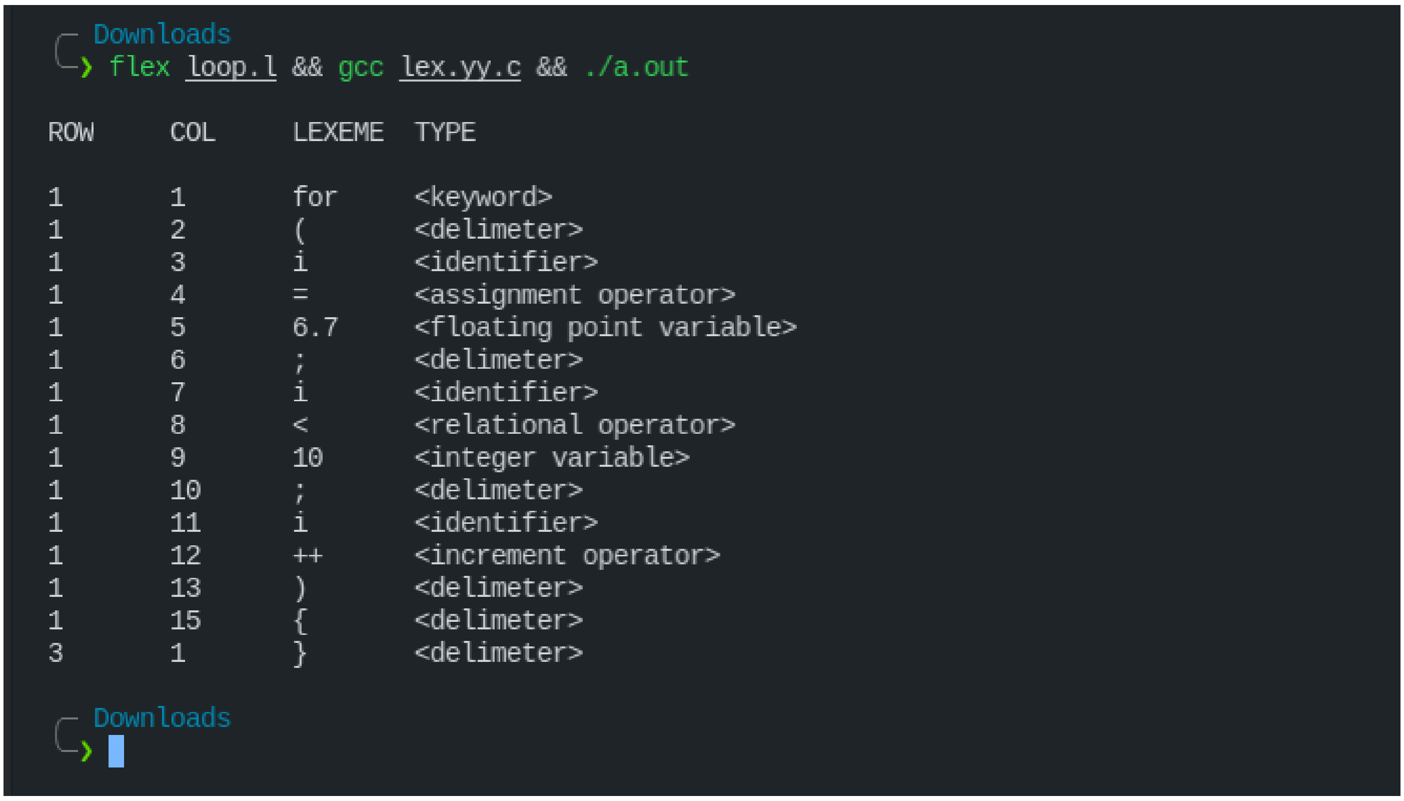
I/O - Screenshot

1.

INPUT

```
for(i=6.7;i<10;i++) {


}
```

OUTPUT



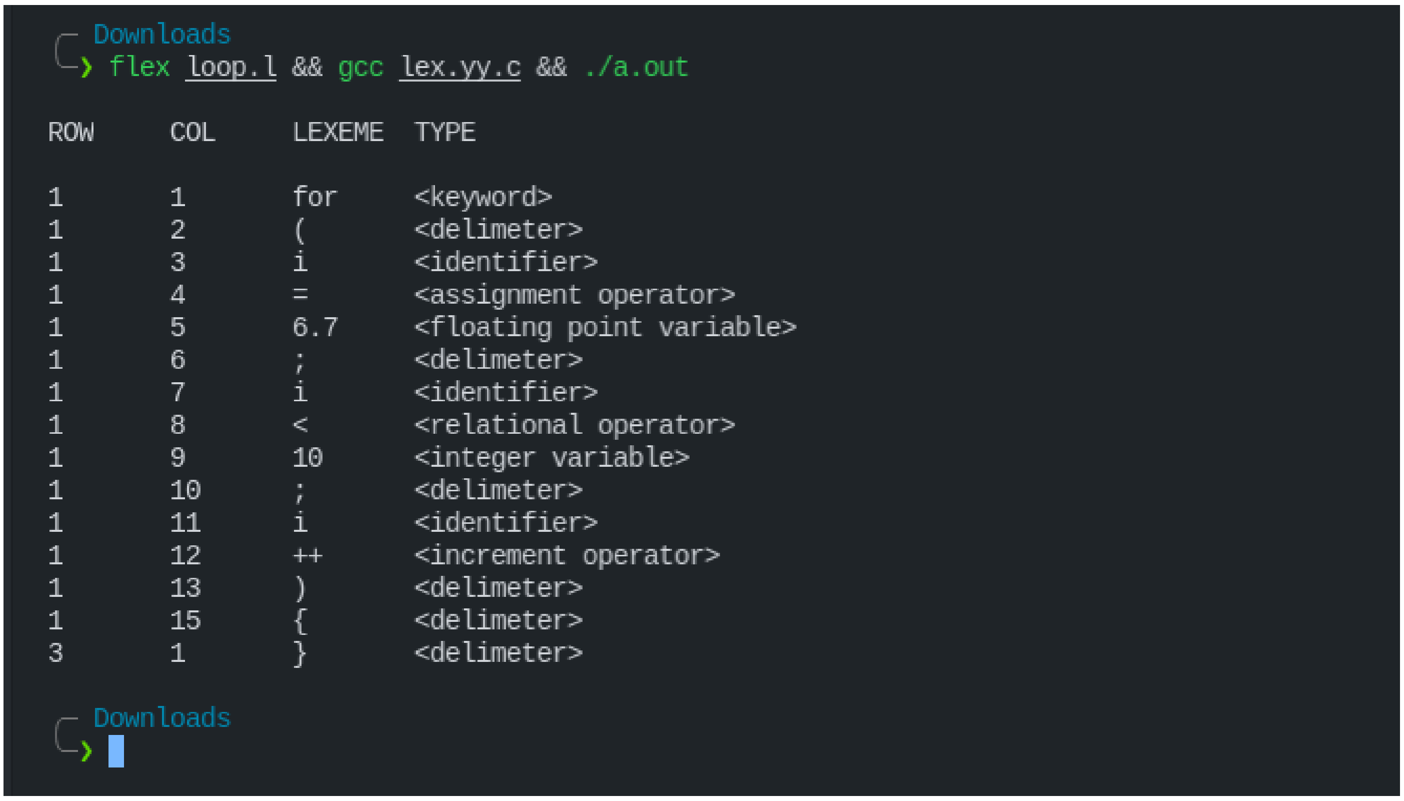
```
╭─ Downloads
╰─› flex loop.l && gcc lex.yy.c && ./a.out

ROW     COL     LEXEME  TYPE

1       1       for     <keyword>
1       2       (       <delimeter>
1       3       i       <identifier>
1       4       =       <assignment operator>
1       5       6.7     <floating point variable>
1       6       ;       <delimeter>
1       7       i       <identifier>
1       8       <       <relational operator>
1       9       10      <integer variable>
1       10      ;       <delimeter>
1       11      i       <identifier>
1       12      ++      <increment operator>
1       13      )       <delimeter>
1       15      {       <delimeter>
3       1       }       <delimeter>

╭─ Downloads
╰─›
```

2.

INPUT

```
for(ib4=6.7;i<10;4Bx++) {

}
```

OUTPUT



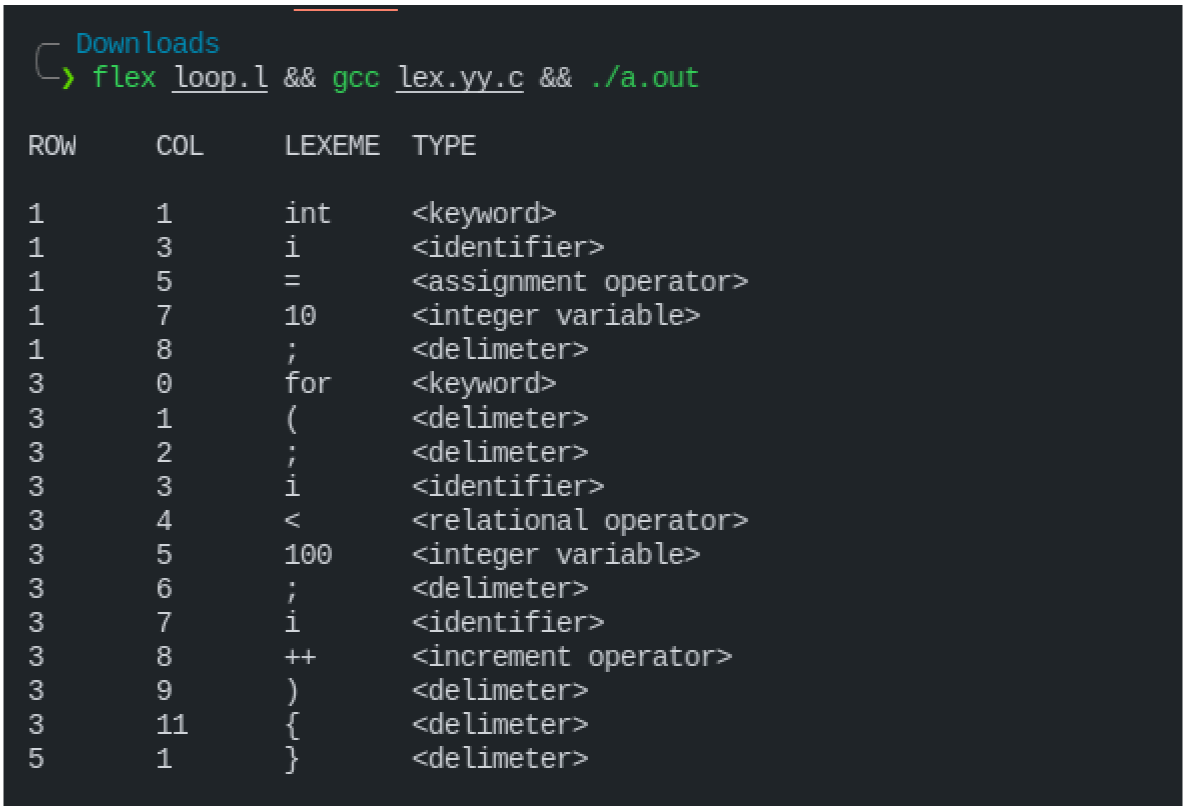
```
╭─ Downloads
╰─❯ flex loop.l && gcc lex.yy.c && ./a.out

ROW     COL     LEXEME  TYPE

1       1       for     <keyword>
1       2       (       <delimeter>
1       3       i       <identifier>
1       4       =       <assignment operator>
1       5       6.7     <floating point variable>
1       6       ;       <delimeter>
1       7       i       <identifier>
1       8       <       <relational operator>
1       9       10      <integer variable>
1       10      ;       <delimeter>
1       11      i       <identifier>
1       12      ++      <increment operator>
1       13      )       <delimeter>
1       15      {       <delimeter>
3       1       }       <delimeter>

╭─ Downloads
╰─❯
```

3.

INPUT

```
int i = 10;
for(;i<100;i++) {

}
```

OUTPUT



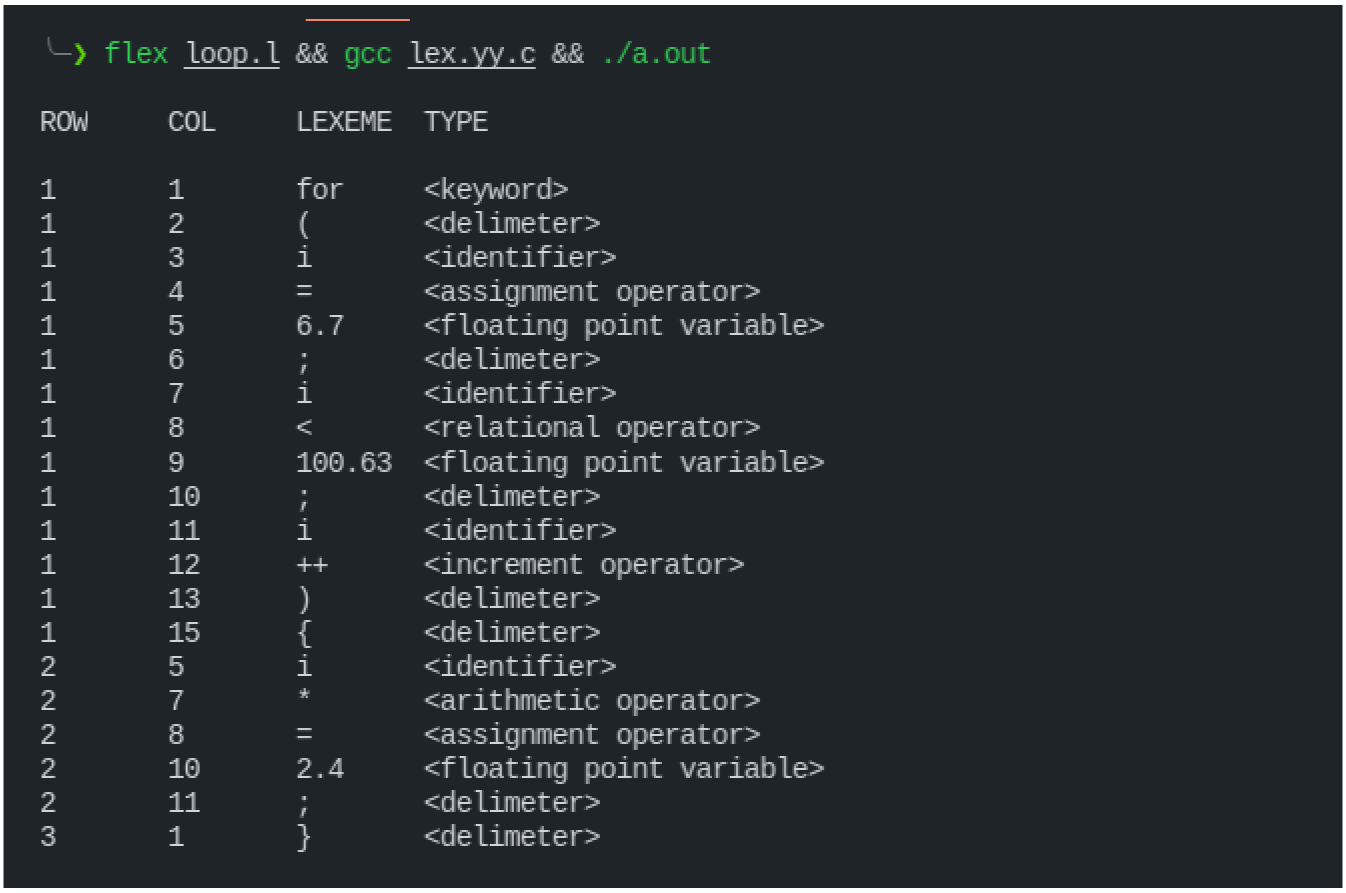
```
╭─ Downloads
╰─❯ flex loop.l && gcc lex.yy.c && ./a.out

ROW     COL     LEXEME  TYPE

1       1       int     <keyword>
1       3       i       <identifier>
1       5       =       <assignment operator>
1       7       10      <integer variable>
1       8       ;       <delimeter>
3       0       for     <keyword>
3       1       (       <delimeter>
3       2       ;       <delimeter>
3       3       i       <identifier>
3       4       <       <relational operator>
3       5       100     <integer variable>
3       6       ;       <delimeter>
3       7       i       <identifier>
3       8       ++      <increment operator>
3       9       )       <delimeter>
3       11      {       <delimeter>
5       1       }       <delimeter>
```

4.

INPUT

```
for(i=6.7;i<100.63;i++) {
    i *= 2.4;
}
```

OUTPUT



```
╰─❯ flex loop.l && gcc lex.yy.c && ./a.out

ROW     COL     LEXEME  TYPE

1       1       for     <keyword>
1       2       (       <delimeter>
1       3       i       <identifier>
1       4       =       <assignment operator>
1       5       6.7     <floating point variable>
1       6       ;       <delimeter>
1       7       i       <identifier>
1       8       <       <relational operator>
1       9       100.63  <floating point variable>
1       10      ;       <delimeter>
1       11      i       <identifier>
1       12      ++      <increment operator>
1       13      )       <delimeter>
1       15      {       <delimeter>
2       5       i       <identifier>
2       7       *       <arithmetic operator>
2       8       =       <assignment operator>
2       10      2.4     <floating point variable>
2       11      ;       <delimeter>
3       1       }       <delimeter>
```