Assignment 2
Lex Implementation

Note: All code must be compilable and executable on Windows systems.
The assignments will be checked on a windows System. Code that does not compile will receive 0 marks. There will be no evaluations. C++ and JAVA will be accepted.

In the previous phase you designed a lexical analyzer for JAVA--. Your task now is to implement it.

The Lex shall take a text file containing JAVA-- source code as input, and shall generate two text files: the first file shall contain the sequence of token-lexeme pairs, and the second shall list down all the identifiers found (Symbol Table).

Following is a sample program written in JAVA--.

```
int numPrint (int num, int length)
{
        int i, j, first, temp;
        char a;
        a <- 'x';
        jOut( "enter number");
        jIn (i);
        jOut(i);
        i <- length;
        while (i > 0)
        {
                first<- 0;              /*this line contains a comment*/
                j <-1;
                while (j < i)
                {
                        jOut( j);
                        j <- j + 1;
                }
                /* this is a comment */
                i<- i - 1;
                /*This is a
                Multiline
                Comment*/
        }
        jOut ("temp is ");
         jOut( temp);
        return i;
}
```

The language contains the following elements:
data types: int char
Keywords: if else while return jIn jOut
arithmetic operators: + - * /
relational operators: < <= > >= == !=
comments: /* enclose comment in */
identifier: a letter followed by any number of letters or digits
numeric constants: only integers
literal constants: a letter enclosed in single quotes
strings: no need to store as variables, only used in print statements
parenthesis, braces, square brackets
assignment operator <-

semi colon
colon
comma

Requirements:

- A test file that successfully runs your lex. Name this test.cmm
- Complete code of your Lex, which will generate the following text files:
- words.txt: generated sequence of all token-lexeme pairs encountered in the given .cm file
- symboltable.txt: the names of all the program identifiers

Like any good compiler, your lex must print appropriate and helpful errors when incorrect programs are provided to it.

Do not hardcode filenames, directory paths, or anything else. If your code cannot be evaluated on machines other than your own, it will not be evaluated.

Interface:
Make a main class that takes filename as input and run your lex on that file.

Make sure you follow all given instructions, and name the files as instructed.

Submission Instructions:
Submit files unzipped
Do not submit executables.
This deliverable will be marked without an evaluation, so make sure the code is compilable **ON WINDOWS**.
There will be zero tolerance for plagiarism. Your assignments will be checked far more thoroughly than you are anticipating. Once detected, no appeals for removal of plagiarism will be entertained.