Write a program which contains a lexical analyzer for the ZinC programming language described in the document posted on Blackboard. The input to the program is a ZinC program. The output will be a sequence of <lexeme, token> pairs. You may to write your lexical analyzer using lex or flex (suggested). or you can hand code your lexical analyzer with a "reasonable" programming language (Java, C, Ada, or Python.) As you encounter identifiers, your lexical analyzer should search your symbol table for a match, and if the identifier is not present, insert the lexeme in the symbol table along with the line number of the line it was first encountered. How you handle reserved words (like **if, while,** etc.) is up to you. You can preload the symbol table with them, OR you can have the lexical analyzer handle them as a special case. When the lexical analysis is finished, you should print out the contents of the symbol table you generated.

For example if your program file starts:

```
1 program
2   var N : integer;
3   var SQRT : integer;
4 begin
5   N := readInt;
```

Then your output would be:

```
<"program", PROGRAM>
<"var",VAR>
<"N", ident>
<":", COLON>
<"integer",INT>
<";",SC>
```

Your program should be able to determine if an input source file (such as myprog.znc) has been passed as a shell argument. If no valid argument is given then a usage message should be printed and the use allowed to input the filname of a ZinC language program. Output should go to standard output.

Turn in your source files via Blackboard.