CS323
Assignment 2
All sections: by 11/15 (Friday), 11:59 pm

The second assignment is to write a syntax analyzer. You may use any top-down parser such as a RDP, a predictive recursive descent parser or a table driven predictive parser.
Hence, your assignment consists of the following tasks:
1. Rewrite the grammar provided to remove any left recursion
   (Also, use left factorization if necessary)
2. Use the **lexer()** generated in the assignment 1 to get the tokens
3. **The parser should print to an output file the <u>tokens, lexemes</u> and the <u>production rules used;</u>**
   That is, <u>**first, write the token and lexeme found**</u>
   Then, <u>**print out all productions rules**</u> used for analyzing this token
   Note: - a simple way to do it is to have a "print statement" at the beginning of each function that will print the production rule.
   - It would be a good idea to have a "switch" with the "print statement" so that you can turn it on or off.
4. **Error handling:** if a syntax error occurs, your parser should generate a meaningful error message, such as token, lexeme, line number, and error type etc.
Then, your program may exit or you may continue for further analysis.
<u>The bottom line is that your program must be able to parse the entire program if it is syntactically correct.</u>
<u>5. Turn in your assignment according to the specifications given in the project outline</u>

Example
Assume we have the following statement
> ….more ….
> a = b + c;
> …. more ….

<u>One possible output would be as follows</u>:
…. more….
Token: Identifier   Lexeme: a
   <Statement> -> <Assign>
   <Assign> -> <Identifier> = <Expression> ;
Token: Operator   Lexeme: =
Token: Identifier   Lexeme: b
   <Expression> -> <Term> <Expression Prime>
   <Term> -> <Factor> <Term Prime>
   <Factor> -> <Identifier>
Token: Operator   Lexeme: +
   <Term Prime> -> ε
   <Expression Prime> -> + <Term> <Expression Prime>
Token: Identifier   Lexeme: c
   <Term> -> <Factor> <Term Prime>
   <Factor> -> <Identifier>
Token: Separator   Lexeme: ;
   <Term Prime> -> ε
   <Expression Prime> -> ε
…. more…..