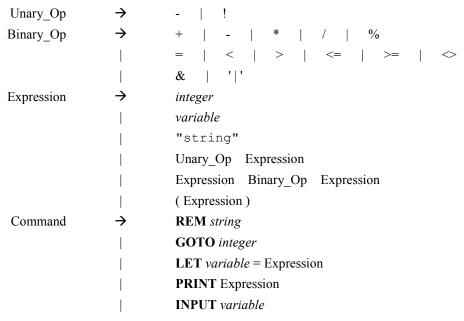
CS383 Assignment 2

Instructions:

- Submit all your answers in hard copies.
- Please submit source codes for question 5 and question 7 to <u>alex_wanq@126.com</u>.
- This assignment is released on 11/11/2011, and due 11/16/2011.
- Please remember to include your name and student ID on all copies.
- 1. Rewrite the productions for *Identifier* and *Float*'s non-terminals as right and left regular grammars.
- 2. Develop 8 test cases for Clite's *Lexer*. Explain the purpose of each test case. Run the test cases and hand in the output as well as the test cases in the following format.

	-		
Input	Expected Output	Actual Output	Pass/Fail
bool i;	TokenType.Bool	(Depends on your	If actual output
	TokenType.Identifier	running result.)	matches the
	TokenType.Semicolon		expectation, then
			pass, otherwise fail.

- 3. Write down the equivalent concrete grammar in BNF for the following EBNF.
 - a) $A \rightarrow x (y|z)$
 - b) $A \rightarrow \{x\}+\{[y|z]\}$
- 4. Design a DFSA that accept all the key words in Clite. Show the moves made using your DFSA for keywords in consuming the following input strings: a) while, b) float.
- 5. Refer to the concrete syntax of BASIC language in the below BNF:



Draw the left dependency graph of BASIC. Define the abstract syntax for BASIC. Implement a recursive descent parser for this grammar in Java. You can assume a class Token which includes Token.integer, Token.variable and Token.str_literal.

- 6. Exercise 4.3, page 99.
- 7. Implement the overloading of operator * in C++, to do matrix multiplication. The matrix can be expressed as a two dimensional array, like m[5][9], which equals to $\begin{bmatrix} 1 & \cdots & 9 \\ \vdots & \ddots & \vdots \\ 5 & \cdots & \end{bmatrix}$ with 5 rows and 9 columns. Write a test program to test the overloaded operator.
- 8. Exercise 4.6, page 99.
- 9. For the following snippet of C++ codes.
 - a) Using static scoping, write down the symbol table stacks at line 5, 12 and 20.
 - b) Using dynamic scoping, write down the symbol table stacks at line 5 for call histories:
 - \rightarrow main(28) \rightarrow A(19) \rightarrow B(11) \rightarrow C
 - ightharpoonup main(28) \rightarrow B(11) \rightarrow C

```
    int x, y, z;
    void C(int var1)

3. {
4.
    int m, n,l;
5.
    x = 2*var1;
6.
     x = x + 1;
7.
8. }
9. void B(int var2, int var3)
10. {
11. C(y);
12. C(var3);
13. x = 300;
14.
15. }
16. void A(int var4, int var5,int var6)
17. {
       float z,w,u;
18.
      B(x,y);
19.
20.
       C(w);
21.
       z = 90;
22.
23. }
24. void main(string[] args)
25. {
26.
       int a, b, c;
27.
       x = 5; a = 3; b = 2; c = 1;
28
       A(a,b,c);
29.
       B(x,y);
30.
       C(z);
31.
32 }
```