CSCI 330 Homework #2

Due Friday, February 16 @ 11:59PM

```
<assign> -> <id> = <expr>
<id> -> A | B | C
<expr> -> <expr> + <term> | <term>
<term> -> <term> * <factor> | <factor>
<factor> -> ( <expr> ) | <id>
```

1. [10pts] Rewrite the BNF grammar above to give + precedence over * and force + to be right associative.

2. [10pts] Using the grammar above, show a parse tree and a leftmost derivation for each of the following statements:

```
a.  A = ( A + B ) * C
b.  A = B * ( C * ( A + B ) )
```

A leftmost derivation of A = B + C would be:

<assign>
<id> = <expr>
A = <expr>
A = <expr> + <term>
A = <term> + <term>
A = <factor> + <term>
A = <id> + <term>
A = B + <term>
A = B + <factor>
A = B + <id>
A = B + C

3. [5pts] Rewrite the BNF grammar above in EBNF

4. [10pts] Prove that the following grammar is ambiguous:

```
<S> -> <A>
<A> -> <A> + <A> + <A> | <id>
<id> -> a | b | c
```

Just show me two parse trees that map to the same statement

5. [10pts] Consider the following grammar:

```
<S> -> <A> a <B> b
<A> -> <A> b | b
<B> -> a <B> | a
```

Which of the following sentences are in the language generated by this grammar?

a. baab
b. bbbab
c. bbaaaaa
d. bbaab
e. babab

Determine the rules of this grammar by analyzing it.
From this you should be able to verify whether sentences are valid.

HINT: it must start with one or more b's

6. [10pts] Write a grammar for the language consisting of strings that have *n* copies of the letter **a** followed by <u>the same number</u> of copies of the letter **b**, where *n* > 0.

7. [5pts] Write an attribute grammar whose BNF basis is the grammar below but whose language rules are as follows: Data types cannot be mixed in expressions, but assignment statements need not have the same types on both sides of the assignment operator.

Write the SEMANTIC rules for determining the type from a variable
HINT: look at the slides r.e. LOOKUP

```
<assign> -> <var> = <expr>
<expr> -> <var>[2] + <var>[3] | <var>
<var> -> A | B | C
```

Write the SEMANTIC rule for determining type match/mismatch for expressions

BONUS: If you don't need to check a type, you don't need to specify a semantic rule

8. [10pts] Compute the weakest precondition for each of the following:

a. *a = 2 * (b – 1) – 1*
{a > 0}
b. *a = 2 * b + 1;*
b = a – 3;
{b < 0}

Start from the bottom and work your way up.

e.g. suppose we have c = b + 2, { b > 2 }

$$\frac{\{ b > 2 \} \mid b = c + 2}{c + 2 > 2}$$

therefore, { c > 4 }