CptS 355
Homework 3

8th February 2005

Due date: Friday, February 25, 2004 at the start of class
Instructions: All answers except those asking for parse trees must be **typewritten** (or in modern terminology word-processed)**.** Parse trees may be handwritten but strive for neatness approaching that of typed answers.

1. This question refers to the following grammar, G:

```
<a> -> <i> = <e>
<i> -> A | B | C | D
<e> -> <e> # <t> | <t>
<t> -> <t> @ <f> | <f>
<f> -> %<e>$ | <i>
```

What are the **terminal** symbols of this grammar?

What are the **non-terminal** symbols of this grammar?

What is the start symbol of this grammar?

Is the grammar ambiguous?

Write a new set of rules, using the same terminal and non-terminal symbols, and generating the same language, but giving # precedence over @ and making @ right associative.

Draw parse trees for C=A@B#B and D=C@D@A#B using grammar G and using your grammar.

2. Write a grammar for lisp-like lists consisting of a's, b's, and nested parenthesized sublists to arbitrary depth. Draw derivation trees according to your grammar for the three lists `(a b), (a (a a) a), (a () (b (a))).`

4. What is the weakest precondition of each of the following assignment statements and postconditions? (note: these problems are not related to one another – each is a separate problem)

```
a) a = 4 * b - 4 {a==0}
b) b = (c mod 3) + 1 {b==1}
c) a = b + c {a==5 or a<0}
d) c = 12 {b+c==10}
e) c = 4+7 {a>5} // this is not a typo!
f) d = 35 {d==20} // this is not a typo either
```

5. What is the weakest precondition of each of the following sequences of assignment statements followed by post-conditions?

```
a) a = 3*b + 4; b = a-4 {b<0}
b) a = n*a ; b = a+b {b>10}
```

6. Consider the following program and postcondition:

```
i = k+1;
product = 1;
while (i<=n) do
   product = product * i;
   i = i+1
end
{product = (n!/k!)}
```

You may assume that all values are integers and that `k` and `n` are positive integers with `k<n`. Determine an invariant for the while statment and show that

a. `(I and not i<=n) => (product==n!/k!)`
b. `(I and i<=n) => wp(product=product*i; i=i+1, {I})`

Determine a precondition, P, for the program as a whole and show that
c. `P => wp(i=k+1; product=1, {I})`

When you have done all of these things you will have shown that

```
{P}
i = k+1;
product = 1;
{I}
while (i<=n) do
   {I and i<=n}
   product = product*i;
   i = i+1
   {I}
end
{I and not i<=n}
{product == n!/k!}
```

is a correctly asserted program using your assertions P and I.

7. Refer to Chapter 15, section 3. What is a most general unifier (*mgu*) for the following equation:

$f(x, g(x), h(g(x))) = f(a, y, h(z))$

Here, f, g, and h are function symbols, a is a constant, and x, y, and z are variables as explained in 15.3.