Natural Language Processing (JHU 601.465/665)
Answers to "Semantics" practice problems

```
1. (a) %g %x big(x), g(x)
       where % = lambda

       A "smartass answer" is %g %x big(x), hat(x),
       which ignores g.

   (b) first blank:  E%x 2(x)    (where E%="there exists")
       second blank: %x 1(x), 2(x)

   (c) first blank:  %g %x big(x),g(x)
       second blank: %g %x red(x),g(x)
       third blank:  %g E%x g(x)

2. (a) S: =>
       NP: =>
       VP: =>

   (b) every politician: lambda f . forall p . politician(p) => f(p)
       every: lambda g . lambda f . forall p . g(p) => f(p)

   (c) i. True.  There is a forall or "generic" quantifier over
          "babies."  See part iii. below.

       ii. False.  Wouldn't make sense.  The quantifier
           is in the right place where it is.

       iii. True.  The second argument of kisses should
            not be a predicate, like "babies".  It should be
            a particular entity who is actually being kissed, like
            "Stewie."

            When we say "Barack kisses babies," it should mean
            something like
               most(lambda b . baby(b), lambda b . kisses(Barack,b))
            so that b refers to each individual baby in turn.
            (The actual quantifier isn't "most" but something
            more like "typically," known as a "generic quantifier.")

       iv. False.  There is nothing WRONG with
           writing kisses(x,y) in the semantics, on the
           assumption that kisses is DEFINED to be a
           function something like
              lambda x. lambda y.
                 exists e . act(e,kissing), time(e,present),
                 kisser(e,x), kissee(e,y)
            so that kisses(x,y) MEANS
                 exists e . act(e,kissing), time(e,present),
                 kisser(e,x), kissee(e,y)

           You are free to replace kisses(x,y) with its definition
           (this is like inlining a function call), but there is
           nothing WRONG with writing kisses(x,y).

           In fact, writing just kisser(x), kissee(y) would make
           things worse.  That leaves out the event variable e, which
           is the only thing ensuring that the kisser and the kissee
           are involved in the same kiss.

       v. False.  We can of course decide whether the first
          argument of kisses is the kisser or the kissee,
          but the tree as shown is completely consistent in
          assuming that the first argument is the kisser.
```

vi. True. This is related to i. and iii. When there are multiple quantifiers, often there are multiple semantic interpretations, having to do with the relative order of those quantifiers. (Remember: "A woman has a baby every 15 minutes.")

In this case, there is an "obvious" meaning: for every politician, he/she will kiss each typical baby in his context. Here, "forall" scopes over "generic."

But there is another meaning where "generic" scopes over "forall", meaning that each typical baby is kissed by every politician:

> Babies are kept safe by society. Mothers feed babies. Schools educate babies. And every politician kisses babies, for a baby who is not kissed by every politician won't grow up.

Notice that this meaning goes with a slightly different intonation when you read the sentence.

(d) forall p. (politician(p), met(we,p)) => kisses(p,babies))

```
(e)                    S
                    /     \
                  /         \
                NP           VP
               /  \         /  \
            Det    N       V    NP
           every  / \   kisses  babies
                 N  CP/NP
         politician /  \
                   C    S/NP
                  that  /  \
                       NP  VP/NP
                       we   / \
                           V   NP/NP
                          met    e
```

(f) (N politician that we met): lambda p. politician(p), met(we,p)
(N politician): lambda p. politician(p)
(CP/NP that we met): lambda g. lambda p. g(p), met(we,p)
(S/NP we met): lambda r. met(we,r)
(C that): lambda h. lambda g. lambda p. g(p), h(p)