Quiz 13b

1. (3 points) Define the rule substitute. It should work as follows:

;;; Query input:

(substitute (a b a c d) a q ?what)

;;; Query output:

(substitute (a b a c d) a q (q b q c d))

;;; Query input:

(substitute (r s t u) s s ?what)

;;; Query output:

(substitute (r s t u) s s (r s t u))

;;; Query input:

(substitute (s (s) s s) s z ?what)

;;; Query output:

(substitute (s (s) s s) s z (z (s) z z))

1. (4 points) Write the rule for list-ref. (list-ref ?x ?y ?z) is the fact “The element of ?x at index ?y is ?z”. Using numbers in logic programming is tricky, so we will use the following representation for the index/numbers:

0 -> zero, (s 0) -> one, (s (s 0)) -> two, (s (s (s 0))) -> three, and so on.

So instead of

(list-ref (she loves you yeah yeah yeah) 1 ?what)

we use

(list-ref (she loves you yeah yeah yeah) (s 0) ?what)

Here are some examples:

;;; Query input:

(list-ref (she loves you yeah yeah yeah) (s (s 0)) ?what)

;;; Query results:

(list-ref (she loves you yeah yeah yeah) (s (s 0)) you)

;;; Query input:

(list-ref (she loves you yeah yeah yeah) ?num yeah)

;;; Query results:

(list-ref (she loves you yeah yeah yeah) (s (s (s (s (s 0))))) yeah)

(list-ref (she loves you yeah yeah yeah) (s (s (s 0))) yeah)

(list-ref (she loves you yeah yeah yeah) (s (s (s (s 0)))) yeah)

Hint: If you match a number to (s ?x), ?x will be that number minus 1.

1. (3 points) Recall the code for the reverse rule:

(assert! (rule (reverse (?a . ?x) ?y)  
 (and (reverse ?x ?z)  
 (append ?z (?a) ?y) )))  
  
(assert! (reverse () ()))

This rule cannot “run backwards”. For example, the below query results in an infinite loop.

;;; Query input:

(reverse ?what (a b c))

Why does this infinite loop? Be as specific as possible (trace through the steps!).