Quiz 13a

1. (2 points) We’re trying to use logic programming to solve the following puzzle:

Both Andy and Charlie are taller than Bob, and Dan is taller than both Andy and Charlie. Who is the tallest? (Assume that there is no ambiguity – we can deduce who the tallest person is.)

We add the following facts:

(assert! (taller andy bob))

(assert! (taller charlie bob))

(assert! (taller dan andy))

(assert! (taller dan charlie))

Now let’s actually solve the puzzle. The first thing that comes to mind is to say that nobody is taller than the tallest person: (not (taller ?other ?tallest)). Unfortunately, this doesn’t work. Fix the query. You can assume that member has been written, and that we only consider Andy, Bob, Charlie, and Dan (this is a hint!). An explanation may earn partial credit, but is not required.

Another hint: not succeeds if for **every** possible binding, the “argument” to the not would fail.

1. (3 points)  Write a rule for length. It should work as shown below. Additionally, you can assume you have a rule plus-one already written.

;;; Query input:                  ;;; Query input:

(length (a b c) ?x)               (plus-one 4 ?what)

;; Query results:                 ;;; Query results:

(length (a b c) 3)                (plus-one 4 5)

;;; Query input:                  ;;; Query input:

(length (a (b)) ?x)               (plus-one ?what 3)

;;; Query results:                ;;; Query results:

(length (a (b)) 2)                (plus-one 2 3)

1. (1+2+2 points) Louis Reasoner takes a look at the definition of list?

(assert! (rule (list? ())))

(assert! (rule (list? (?a . ?b))

(list? ?b)))

“It’s all wrong! The recursive rule is for a list with at least two elements, and the base case is for a list with no elements. There’s no case for a list with only one element!”

1. Explain what Louis has misunderstood.
2. To prove that the rule works for one element lists, show all of the (successful) unifications that allow the logic programming language to derive the fact (list? (one)).
3. Unfortunately, Louis has already added a new rule to deal with lists with one element:

(assert! (rule (list? (?x))))

Now, Ben Bitdiddle tries to use the list? rule, and sees the following:

;;; Query input:

(list? (is this a list))

;;; Query results:

(list? (is this a list))

(list? (is this a list))

Explain why (list? (is this a list)) is printed out twice.