Quiz 3a

1. (3 points) Which of the following can always be done in Θ(n) time or better with

respect to input size? (There may be zero, one, or multiple correct choices.)

\_\_\_ Given a keyring and a lock, figure out if one of the keys opens the

lock.

\_\_\_ Given a keyring and a set of locks, figure out which key

opens which lock.

\_\_\_ Given a set of unlabelled wires and a set of unlabelled light bulbs (all

currently lit), figure out which wire goes with which light bulb.

\_\_\_ Figure out how many fashionable outfits are possible (by trying each

outfit on and examining yourself in the mirror), given a number of

shirts and pants. Input size is the sum of the number of shirts and

the number of pants.

\_\_\_ Given a keyring and a set of locks, figure out if any of the keys

can open any of the locks.

2. (3 points) Circle the procedures below (if any) that generate an iterative process.

Don't circle the ones (if any) that generate a recursive process.

(define (triangle n)

(define (help cnt)

(if (= cnt n)

cnt

(+ cnt (help (+ cnt 1)))))

(help 1))

(define (rep fn n x)

(if (= n 0)

x

(rep fn (- n 1) (fn x))))

(define (pigl wd)

(if (vowel? (first wd))

(word wd 'ay)

(pigl (word (bf wd) (first wd)))))

(define (fib n)

(define (help a b n)

(if (= n 0)

a

(help b (+ a b) (- n 1))))

(help 1 1 n))

3. (4 points) The following procedure takes three arguments: an item, a sentence, and a

two-argument predicate. It returns the first element in the sentence which is ``like'' the item, where two things are considered ``like'' each other if the predicate returns true when applied to them. If no element of the sentence is like the item, false is returned. Assume the predicate takes constant time.

(define (find-like item sent pred)

(cond ((empty? sent) #f)

((pred item (first sent)) (first sent))

(else (find-like item (bf sent) pred))))

Does find-like generate a recursive or an iterative process?

What is its running time for a sentence of length N? Θ(\_\_\_)

The following procedure (which uses find-like, from above) takes a sentence and a two-argument predicate, and returns a subset of the given sentence such that no two elements are like each other.

(define (remove-likes sent pred)

(cond ((or (empty? sent) (empty? (bf sent))) sent)

((find-like (first sent) (bf sent) pred)

(remove-likes (bf sent) pred))

(else

(se (first sent) (remove-likes (bf sent) pred)))))

Does remove-likes generate a recursive or an iterative process?

What is its running time for a sentence of length N? Θ(\_\_\_)