Quiz 3a

1. (3 points) What is the order of growth of the running time of the 3 procedures sort, min-sent, remove-one below in terms of n, the length of sent? Explain your answer in the case of sort. Assume that all primitive procedures take constant time.

(define (sort sent)

(if (empty? sent)

sent

(se (min-sent sent) (sort (remove-one (min-sent sent) sent)))))

(define (min-sent sent)

(if (empty? (bf sent))

(first sent)

(min (first sent) (min-sent (bf sent)))))

(define (remove-one elmt sent)

(cond ((empty? (bf sent)) ‘())

((equal? elmt (first sent)) (bf sent))

(else (se (first sent) (remove-one elmt (bf sent))))))

1. (2 points) What is the order of growth of the following procedure in terms of n, and what kind of process does it generate (recursive or iterative)? Assume all primitive procedures take constant time. We may give partial credit for explanations.

(define (binary n)

(define (loop num)

(if (= num 0)

‘()

(se (loop (quotient num 2)) (remainder num 2))))

(loop n))

1. (1+1+3 points) Consider the reverse procedure, which reverses its input sentence:

> (reverse ‘(anne i vote more cars race rome to vienna))

(vienna to rome race cars more vote i anne)

(define (reverse sent)

(if (empty? sent)

‘()

(se (last sent) (reverse (bl sent)))))

1. Does this generate a recursive or iterative process?

Suppose you know that last and bl take Θ(n) time, where n is the length of the sentence, and all other word and sentence functions that are provided for you take Θ(1) time (this isn’t true, but assume it is for the rest of this question).

1. What is the order of growth of the running time of the reverse procedure above? Express your answer using Θ notation, in terms of n, the length of sent.
2. We want to make reverse as efficient as possible. We have to at least look at every word in the sentence, so it must take at least linear time. Since we want memory efficiency, we need an iterative process. Write a version of reverse which takes Θ(n) time and generates an iterative process. You can get partial credit by writing an appropriate invariant, but this is **not** required in order to get full credit.