**Part 4: (20 points)**

Suppose that we want to sort a list of elements (e.g. a list of subjectrecs from the previous part – although you can treat this part as independent of the previous part). Here is a procedure for sorting:

```
(define (find-best best todo compare)
  (if (null? todo)
      best
      (if (compare (car todo) best)
          (find-best (car todo) (cdr todo) compare)
          (find-best best (cdr todo) compare))))

(define (remove elt todo same)
   (if (null? todo)
       nil
       (if (same elt (car todo))
           (cdr todo)
           (cons (car todo) (remove elt (cdr todo) same)))))

(define (sort data compare same)
  (let ((trial (find-best (car data) (cdr data) compare)))
     (let ((todo (remove trial data same)))
       (if (null? todo)
           (list trial)
           (cons trial (sort todo compare same))))))
```

For example, to sort our data by increasing subject number, we would evaluate:

```
(sort (get-all-subjectrecs transcript)
      (lambda (x y) (< (subject x) (subject y)))
      eq?)
```

We are going to measure the order of growth in time (as measured by the number of primitive operations in the computation) and in space (as measured by the maximum number of deferred operations – do not count in space the intermediate data structures constructed by the algorithm), measured as a function of the size of data, denoted by n. Assume that the procedures used for compare and same use constant time and space.

For each of the following questions, choose the description from these options that best describes the order of growth of the process. If you select "something else", please state why.

A: constant
B: linear
C: exponential
D: quadratic
E: logarithmic
F: something else

**Question 15.** What is the order of growth in time of the procedure `find-best`?

**Question 16.** What is the order of growth in space of the procedure `find-best`?

**Question 17.** What is the order of growth in time of the procedure `remove`?

**Question 18.** What is the order of growth in space of the procedure `remove`?

**Question 19.** What is the order of growth in time of the procedure `sort`? Remember to include the effect of `find-best` and `remove`.

**Question 20.** What is the order of growth in space of the procedure `sort`? Remember to include the effect of `find-best` and `remove`.

**Part 4: (15 points)**

Here are two procedures to reverse the elements of a list:

```
(define (rev lst)
  (define (help done todo)
    (if (null? todo)
        done
        (help (cons (car todo) done) (cdr todo))))
  (help '() lst))

(define (rev1 lst)
  (if (null? lst)
      '()
      (append (rev1 (cdr lst))
              (list (car lst)))))

(define (append l1 l2)
  (if (null? l1)
      l2
      (cons (car l1) (append (cdr l1) l2))))
```

We would like to measure the order of growth in time (as measured by the number of primitive operations in the computation) and in space (as measured by the maximum number of deferred operations – do not count in space the intermediate data structures constructed by the algorithm).

For each of the following questions, choose the description from these options that best describes the order of growth of the process.

A: constant
B: linear
C: exponential
D: quadratic
E: logarithmic
F: something else

**Question 14.** What is the order of growth in time of the procedure `rev`, when applied to a list of `n` elements?

**Question 15.** What is the order of growth in space of the procedure `rev`, when applied to a list of `n` elements?

**Question 16.** What is the order of growth in time of the procedure `rev1`, when applied to a list of `n` elements?

**Question 17.** What is the order of growth in space of the procedure `rev1`, when applied to a list of `n` elements?