Suppose that you want to sort a list of elements. Here is a procedure for sorting:

```scheme
(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)
      '()
      (if (same elt (car 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))))))
```

To sort a list of numbers called my-list in ascending order, for example, we could write:

```scheme
(sort my-list (lambda (x y) (< x y)) =)
```

For each of the following questions choose the letter corresponding to the option that best describes the order of growth of the process in question.

A: constant
B: linear
C: exponential
D: quadratic
E: logarithmic
F: something else
What is the order of growth in time for find-best?  

What is the order of growth in space for find-best?  

What is the order of growth in time for remove?  

What is the order of growth in space for remove?  

What is the order of growth in time for sort?  

What is the order of growth in space for sort?