Exercise 2.64.

The following procedure `list->tree` converts an ordered list to a balanced binary tree. The helper procedure `partial-tree` takes as arguments an integer n and list of at least n elements and constructs a balanced tree containing the first n elements of the list. The result returned by `partial-tree` is a pair (formed with `cons`) whose `car` is the constructed tree and whose `cdr` is the list of elements not included in the tree.

```
(define (list->tree elements)
  (car (partial-tree elements (length elements))))

(define (partial-tree elts n)
  (if (= n 0)
      (cons '() elts)
      (let ((left-size (quotient (- n 1) 2)))
        (let ((left-result (partial-tree elts left-size)))
          (let ((left-tree (car left-result))
                (non-left-elts (cdr left-result))
                (right-size (- n (+ left-size 1))))
            (let ((this-entry (car non-left-elts))
                  (right-result (partial-tree (cdr non-left-elts)
                                              right-size)))
              (let ((right-tree (car right-result))
                    (remaining-elts (cdr right-result)))
                (cons (make-tree this-entry left-tree right-tree)
                      remaining-elts)))))))))
```

a. Write a short paragraph explaining as clearly as you can how `partial-tree` works. Draw the tree produced by `list->tree` for the list `(1 3 5 7 9 11)`.

b. What is the order of growth in the number of steps required by `list->tree` to convert a list of n elements?

Answer.

a. The `partial-tree` procedure works as follows:

- If the number of elements is 0, return a pair whose `car` is `nil` and whose `cdr` is the original list.

- Otherwise, we first seperate the first $\frac{n-1}{2}$ elements from the rest to construct the left balanced subtree and pick the entry element. Then devide elements of its right part from those remainings to build the right balanced subtree. Finally, we return a pair whose `car` is the balanced binary tree we just built and whose `cdr` is a list containing the remaining elements.

Figure 1 shows the tree produced by `list->tree` for the list `(1 3 5 7 9 11)`.

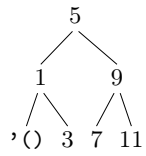

Figure 1. The tree produced by `list->tree` for the list `(1 3 5 7 9 11)`.

b. Note that what we are about to build is a balanced binary tree, so each of these subtrees will be about half the size of the original. Thus, in one step we habe reduced the problem of building a tree of size n to build a tree of size $n/2$. Since the number of steps needed to search a tree of size n grows as $\Theta(\log n)$.