**Objective:** To understand recursion by writing simple recursive solutions.

**To start the lab:** Download and unzip the lab5.zip file from eLearning.

# **Part A:** Recall: We modified the textbook’s ordered list ADT that uses a singly-linked list implementation by adding the \_size, \_tail, \_current, \_previous, and \_currentIndex attributes:



a) What are the base case(s) for the searchHelper that **halt** the while-loop of the non-recursive search codc?

b) What are the recursive case(s) for the searchHelper that replaces the while-loop of the non-recursive search codc?

c) Complete the recursive searchHelper function in the search method of our OrderedList class in ordered\_linked\_list.py. Run the ordered\_linked\_list.py test code at the bottom of the class, or test it with the listTester.py program.

# **Part B:** Recall that Lecture 7 and Section 6.6 discussed a very “non-intuitive”, but powerful list/array-based approach to implement a priority queue, call a binary heap. The list/array is used to store a *complete binary tree* (a full tree with any additional leaves as far left as possible) with the items being arranges by *heap-order property*, i.e., each node is  either of its children. An example of a *min* heap “viewed” an a complete binary tree would be:

Recall the General Idea of insert(newItem):



1. append newItem to the end of the list (easy to do, but violates heap-order property)
2. restore the heap-order property by repeatedly swapping the newItem with its parent until it *percolates up* to the correct spot

Recall the General Idea of delMin():

1. remember the minimum value so it can be returned later (easy to find - at index 1)
2. copy the last item in the list to the root, delete it from the right end, decrement size
3. restore the heap-order property by repeatedly swapping this item with its smallest child until it *percolates down* to the correct spot
4. return the minimum value

Originally, we used iteration (i.e., a loop) to percolate up (see percUp) and percolate down (see percDown) the tree. (textbook code below)



**For part B, I want you to complete the recursive** percUpRec and recursive percDownRec methods in binHeap.py. Run the binHeap.py file which has test code at the bottom to test **both** methods.

**After you have correct code for both parts of the lab, submit a lab5.zip containing your code on eLearning.**

**If you do not get done today, then submit it by next week’s lab period.**

(If you have extra time, work on previous labs or homeworks!)