Problem 1. Consider an array of size nine with the numbers in the following order:

$$40, 60, 20, 80, 70, 90, 30, 10, 50.$$

(a) Create the heap using the algorithm described in class. Show the heap as a tree. Show the heap as an array. Exactly how many comparisons did heap creation use?

(b) Start with the heap created in Part (a). Show the *array* after each element sifts down *during the Finish phase*. How many comparisons does each sift use? What is the total number of comparisons *after heap creation*?

Problem 2. For this problem, you may not look at any other *code* or *pseudo-code* (even if it is on the internet), other than what is on our website or in our book. You may discuss general ideas with other people.

Assume `A[1...n]` is a heap, except that the element at index `i` might be too large. For the following parts, you should create a method that inputs `A`, `n`, and `i`, and makes `A` into a heap.

(a)

i. Write pseudo-code to sift up the element at index `i`.

ii. What is the number of comparisons to sift up the element at index `i`. Just give the exact high order term.

(b) The sift up algorithm can be made more efficient, in the worst case, by skipping every other level, i.e., checking the grandparent, and only checking the parent when necessary.

i. Write pseudo-code for this improved version of sift up.

ii. What is the number of comparisons to sift up the element at index `i`. Just give the exact high order term.

iii. Write code for this in any reasonable programming language (such as JAVA, Python, Ruby, R, C, C++, or C#). Turn in the code in your pdf file.

iv. Show the output of your code on the following three inputs:

- A = [21, 20, 19, 18, 11, 10, 9, 8, 7, 6, 5, 4, 16, 2, 1], `n` = 15, and `i` = 13. The value 16 should sift up giving: [21, 20, 19, 18, 11, 16, 9, 8, 7, 6, 5, 4, 10, 2, 1].
- `A` = [15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 16, 2, 1], `n` = 16, and `i` = 13. The value 16 should sift all the way up.
- `A` = [16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1], `n` = 16, and `i` = 13. The value 4 should stay where it is.

Problem 3. Consider the recurrence, from Problem 3 of Homework 3, for the number of comparisons in our recursive version of Modified Insertion Sort (for n even):

$$T(n) \;=\; T(n-2) + \frac{n}{2} + 1, \qquad\qquad T(0) = 0$$

Solve the recurrence using the tree method. Note that the branching factor is not very interesting.