Problem 1. Consider an array of size eight with the numbers in the following order $10, 30, 50, 70, 20, 40, 60, 80$.

(a) What is the array after heap creation? How many comparisons does heap creation use?

(b) Starting from the heap, show the array after each sift operation in the "finish" phase, and state how many comparisons each sift takes. How many comparisons does the algorithm use in total during the "finish" phase?

Problem 2. A *perfect binary tree* is a tree in which every node other than the leaves has exactly two children, and the leaves are all at the same level. We are going to find the exact number of comparisons (in the worst case) for heap creation in a perfect binary tree. We will guess the formula by looking at a few small examples, and then prove it is correct by mathematical induction.

(a) The *height* of a tree is the number of levels of nodes. (A tree consisting of a single root node has height 0. A tree consisting of a root node with some children has height 1. Etc.)

(i) Assume a perfect binary tree has height h. How many nodes does it have? Justify.

(ii) Assume a perfect binary tree has n nodes. What is its height? Justify.

(b) Calculate by hand the exact number of comparisons for perfect binary trees with heights $0, 1, 2, 3, 4$.

(c) We know that the true answer should be approximately $2n$. Find the differences between $2n$ and your calculated values.

(d) Guess a formula for your differences as a function of n.

(e) What formula does that give you for the exact number of comparisons for heap creation as a function of n?

(f) Heap creation can be thought of as a recursive procedure: Create heap for left child of root, create heap for right child of root, and sift the root value down. Write a recurrence for the number of comparisons to create a heap as a function of n.

(g) Use mathematical induction to prove that your formula is a solution to the recurrence.