1. Where in a min heap of integers might the largest element reside? Explain.

It will be a leaf

2. Insert integers $5, 3, 17, 10, 85, 2, 19, 6, 22, 4$ one-by-one into an initially-empty min heap. Re-draw the heap each time an insertion causes one or more swaps.

3. Repeat the previous problem but now use the build-heap algorithm. Redraw the heap each time a call to percolate_down causes one or more swaps.

4. For the binary heap of Exercise 1, show the result of performing four consecutive pop operations. Re-draw the heap after each pop.

5. Use induction to prove that $1 + 2 + 4 + \ldots + 2^h = 2^{h+1} - 1$. Conclude that $2^{h+1} - 1$ is the maximum number of elements that can be stored in a binary heap having height h.

$2^{0+1} - 1 = 1$ hence $P(0)$ is true. We assume $P(k)$ holds for some positive integer k and show it must hold for the $k+1$ integer. $\sum_{i=0}^{k+1} 2^i = \sum_{i=0}^{k} 2^i + 2^{k+1} = 2^{k+1} - 1 + 2^{k+1} = 2^{(k+1)+1} - 1$. which proves $P(k+1)$ is true. By the principle of mathematical induction the result holds for all $h \geq 0$. we thus have in a perfect binary heap of height h we have at exactly $2^{h+1} - 1$ elements.

6. Prove the minimum and maximum number of elements that can be stored in a binary heap that has height h.

In the minimum case, a binary heap with height h has only one node in the lefmost position and every other level i filled where $0 \leq i < h$. Summing the nodes we obtain $\sum_{i=0}^{h-1} 2^i + 1 = 2^h - 1 + 1 = 2^h$. In the maximum case we have a perfect binary heap, problem 5 gives us that the tree will have $2^{h+1} - 1$ nodes.

8. Prove that a binary heap with n nodes has exactly $\lceil \frac{n}{2} \rceil$ leaves.

The last internal node index is $\lfloor \frac{n}{2} \rfloor$. We have $n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$ which are the nodes after the last internal node, the leafs. We have then that a binary heap with n nodes has exactly $\lceil \frac{n}{2} \rceil$ leaves.

9. Given an example which shows that the buildheap algoithm does not work if one begins percolating down with the first internal node, rather than the last internal node.

10. Prove that a binary heap with n elements has height $\lfloor log(n) \rfloor$.

$$\begin{aligned} 2^h \leq & \quad n & \leq 2^{h+1} - 1 \\ 2^h \leq & \quad n & < 2^{h+1} \\ h \leq & \quad log(n) & < h+1 \end{aligned}$$

which by the definition of the floor $h = \lfloor log(n) \rfloor$.

11. Show that there at most $\lceil \frac{n}{2^{h+1}} \rceil$ nodes of height h in any n-element heap.

We prove by induction. Let $P(h)$ be the statement that there are at most $\lceil \frac{n}{2^{h+1}} \rceil$ nodes of height h in any n-element heap. $P(0)$ holds since $\lceil \frac{n}{2^{0+1}} \rceil$ are the number of leaves in the heap, by question 13, which by definition have height 0. We assume $P(k)$ holds for some integer k and show that it must hold for the $k+1$ integer. We make a new heap H_1 which haves all the elements of the original heap H except the leaves, hence it has $n - \lceil \frac{n}{2} \rceil = \lfloor \frac{n}{2} \rfloor$ total elements. The leaves of H_1 will correspond to the nodes of height 1 in the orignal heap. Since H_1 has $\lfloor \frac{n}{2} \rfloor$ total elements it has $\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2} \rceil = \lceil \frac{\frac{n}{2}}{2} \rceil = \lceil \frac{n}{2^2} \rceil$ leaves. We now make a new heap H_2 from H_1 which have all the elements of H_1 except it leaves. H_2 will then have $\lfloor \frac{n}{2} \rfloor - \lceil \frac{n}{2^2} \rceil = \lfloor \frac{n}{2^2} \rfloor$ total elements. Since H_2 has $\lfloor \frac{n}{2^2} \rfloor$ total elements it has $\lceil \frac{\lfloor \frac{n}{2^2} \rfloor}{2} \rceil = \lceil \frac{\frac{n}{2^2}}{2} \rceil = \lceil \frac{n}{2^3} \rceil$ leaves. We do this process k times and by the inductive hyptothesis since there are at most $\lceil \frac{n}{2^{k+1}} \rceil$ nodes of height k in any n-element heap, the heap H_k which have the original Heap H nodes of height k as leaves has $\lceil \frac{n}{2^{k+1}} \rceil$ leaves. We again make a new heap H_{k+1} which will remove the leaves of H_k and has the nodes of height $k+1$ in the original heap as leaves. The heap H_{k+1} has $\lfloor \frac{n}{2^{k+1}} \rfloor$ total elements since $\lfloor \frac{n}{2^k} \rfloor - \lceil \frac{n}{2^{k+1}} \rceil = \lfloor \frac{n}{2^{k+1}} \rfloor$. It then has $\lceil \frac{\lfloor \frac{n}{2^{k+1}} \rfloor}{2} \rceil = \lceil \frac{\frac{n}{2^{k+1}}}{2} \rceil = \lceil \frac{n}{2^{(k+1)+1}} \rceil$ leaves and are the number of nodes of height $k+1$ in the original heap. Hence $P(k+1)$ holds. By the principle of mathematical induction, $P(h)$ holds for all integers $h \geq 0$.

12. Suppose that instead of binary heaps, we wanted to work with ternary heaps. Suggest an appropriate indexing scheme so that a complete tree will yield a contigous sequence.

$$\begin{aligned} left\_child &= 3 \cdot i + 1 \\ middle\_child &= 3 \cdot i + 2 \\ right\_child &= 3 \cdot i + 3 \end{aligned}$$

13. Prove that the worst-case running time of HeapSort is $O(n \cdot log(n))$.

After using using Build-Heap operation which takes $O(n)$ we continously extract elements from the heap until $n = 0$. $\sum_{i=1}^{n} log(i) = O(\int_{1}^{n} log(i) \cdot dx) = O(n \cdot log(n))$.