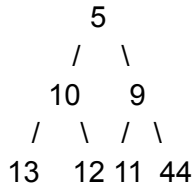
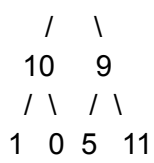
1. What is a heap?
2. How are they different from a binary search tree?
3. True or False: There is no horizontal relationship between a heap?
4. What is the difference between a min heap and max heap?
5. Insert the following values into a min heap and a max heap {8, 15, 12, 25, 1, 14, 37}
6. Is the following a valid min heap?



7. Is the following a valid max heap?



8. What is the time complexity of adding to and removing from a heap?
9. Why are arrays better to use for heaps than LinkedLists?
10. What is a set?
11. True or False: A set in programming is very similar to sets in math?
12. What is the difference between a Set and a Map?
13. What is a hashcode?
14. True or False: Hash codes are always unique.
15. Draw and label the JCF.

Bonus Question that you didn't go over in class (Answer on page 2):

Why should the equals method determine equality using the same state that the hashcode method uses to generate a hash?

Answer:

Because equals and hashcode are related functions. The way that a hashing data structures determines if something is already in the data structure is by computing the hash value, checking the appropriate index, and checking equality if a value already exists in that spot.

Let's say you have the following class

```
public class Employee{

        private String name;
        private String jobTitle;
        private int pay;

        …
}
```

Let's say you determine equality by checking if the two Employees have the same name and jobTitle. If you define your hashcode to use the pay, that means that two Employees with the same name and jobTitles, but different pay (equal employees by our standard) would hash to different values. That's very bad, because now our data structure has a duplicate and that can't happen. Equal objects need to hash to the same value to avoid duplicates, so they should use the exact same state. You can use less, say only the name, but that would lead to more collisions, which would lower your time complexity from O(1) to potentially O(n). Not great.