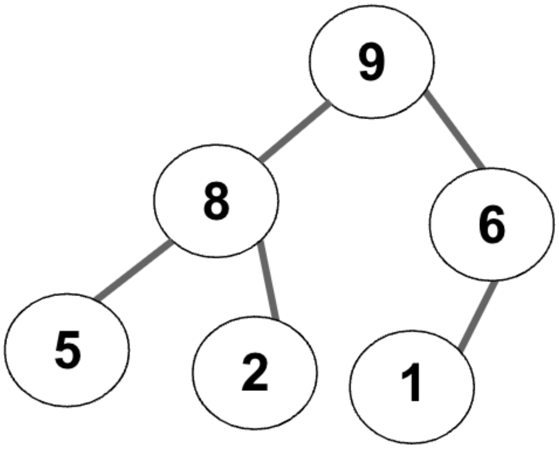
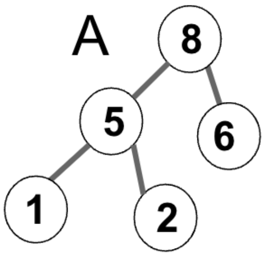
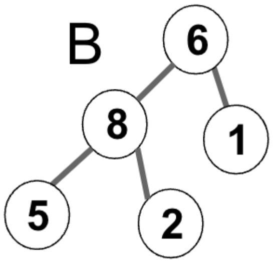
Section 06: Heaps and Hashing

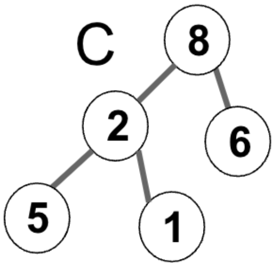
1. Heap Removal

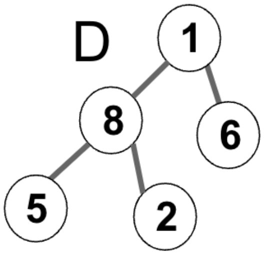
What does the final heap look like after removing the root? ***Select One.***











What are the operations that take place when removing the root in a heap?

1.

2.

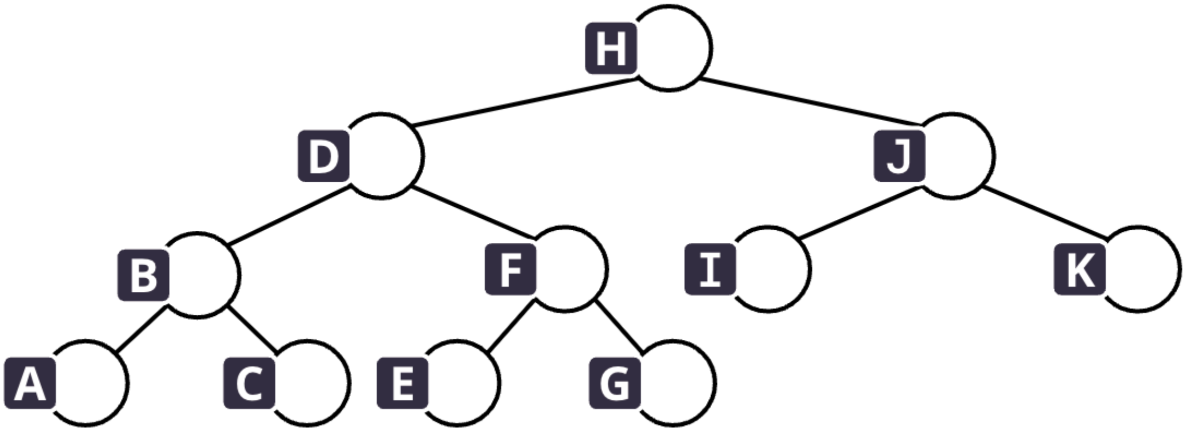
3.

2. Extra-Practice: Min-Heap Values

Assume the integers 1-11 were inserted into the min-heap below in an unknown order.

What are all the possible nodes (A-K) where you could find each of the numbers below?

Explain your reasoning for both the places you could find each number and the places you could not find each number.



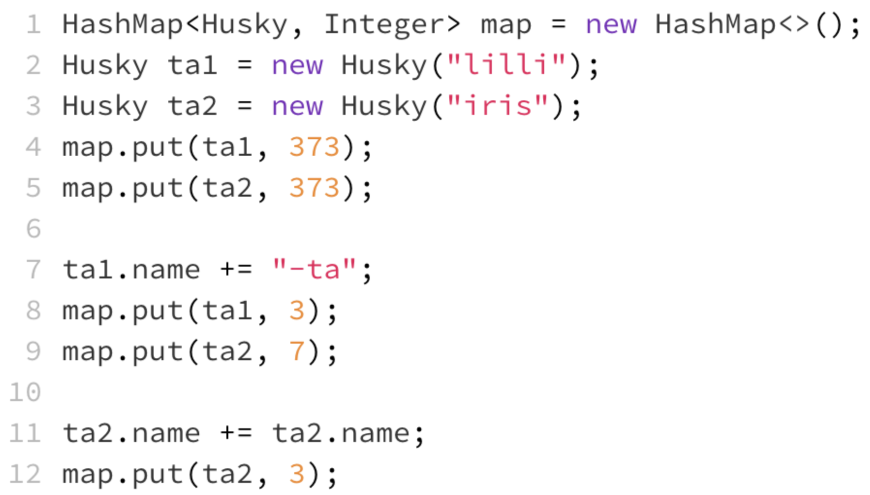
A. **The number 11** ***(the largest value)***

B. **The number 6** ***(the median value)***

C. **The number 3**

3. Hashing!

Please draw the 3 different states of the hashmap as prompted below.



```
 1 HashMap<Husky, Integer> map = new HashMap<>();
 2 Husky ta1 = new Husky("lilli");
 3 Husky ta2 = new Husky("iris");
 4 map.put(ta1, 373);
 5 map.put(ta2, 373);
 6
 7 ta1.name += "-ta";
 8 map.put(ta1, 3);
 9 map.put(ta2, 7);
10
11 ta2.name += ta2.name;
12 map.put(ta2, 3);
```

Draw what the hashmap looks like after lines 1-5 have been run.

Draw what the hashmap looks like after lines 1-9 have been run.

Draw what the final state of the hashmap looks like after lines 1-12 have been run.

What does `map.size()` return and what's the reason why?

4. Extra Practice: Hashing Points

Considering the following code:

```
public class Point {
        public final int x, y;

        public Point(int x, int y) {
                this.x;
                this.y;
        }

        public int hashCode() {               public boolean equals(Object o) {
                return this.x + this.y;               Point other = (Point) o;
        }                                             return this.x == other.x;
}                                             }
```

A. **Check all the points that will collide with `Point(1, 2)` on a hash table with M = 2 buckets.** *Explain why each point does or does not collide by using its hashCode to find the index of its bucket.*

□ `Point(1, 1)`

□ `Point(2, 1)`

□ `Point(3, 1)`

□ `Point(1, 3)`

□ `Point(1, 4)`

□ `Point(1, 5)`

B. Assume `Point(1, 2)` is put into an empty hash table with M = 2 buckets using `Point.hashcode`. **If two points are equal if and only if their x-values are equal, what would calling `contains(Point(1, 3))` on the hash table return?**

Explain your answer by explaining how contains uses both the hashCode and equals methods to search for objects.

A. TRUE
B. FALSE
C. NOT ENOUGH INFORMATION

5. AAC: Design & Implement!

`commonWords(List<String> words)`

- `words`: A long, long list of utterances/speech from many different humans.
    - Ex: {"thanks", "bye", "hi", “good morning”, “thanks”, “hi”, “hello world”, “hi”}
- Returns: The top 50 most common words/phrases a human uses arranged in a **List,** to be used for creating a new page/screen in an AAC.
    - Be sure that you include all necessary information for a button!
    - You should be counting words and sorting them in some way to create a sorted List.

Using HashMaps and/or Heaps, write out an approach in a few English sentences.

What is the runtime of your approach? Think about the different asymptotic cases (best case/worst case/overall case, or what an average experimental case may be).

Considering our proposed recursive 1D Array for the Interface…

What if a HashMap was chosen instead, or another alternative?

How would we integrate commonWords into this 1D Array Interface?