1 Fill in the Blanks

Fill in the following blanks related to min-heaps. Let \( N \) is the number of elements in the min-heap. For the entirety of this question, assume the elements in the min-heap are distinct.

1. `removeMin` has a best case runtime of \( \underline{\text{__________}} \) and a worst case runtime of \( \underline{\text{__________}} \).

2. `insert` has a best case runtime of \( \underline{\text{__________}} \) and a worst case runtime of \( \underline{\text{__________}} \).

3. A \( \underline{\text{__________}} \) or \( \underline{\text{__________}} \) traversal on a min-heap may output the elements in sorted order. Assume there are at least 3 elements in the min-heap.

4. The fourth smallest element in a min-heap with 1000 elements can appear in \( \underline{\text{__________}} \) places in the heap.

5. Given a min-heap with \( 2^N - 1 \) distinct elements, for an element
   - to be on the second level it must be less than \( \underline{\text{__________}} \) element(s) and greater than \( \underline{\text{__________}} \) element(s).
   - to be on the bottommost level it must be less than \( \underline{\text{__________}} \) element(s) and greater than \( \underline{\text{__________}} \) element(s).

*Hint*: A complete binary tree (with a full last-level) has \( 2^N - 1 \) elements, with \( N \) being the number of levels.
2 Heap Mystery

We are given the following array representing a min-heap where each letter represents a **unique** number. Assume the root of the min-heap is at index zero, i.e. A is the root. Note that there is **no** significance of the alphabetical ordering, i.e. just because B precedes C in the alphabet, we do not know if B is less than or greater than C.

Array: [A, B, C, D, E, F, G]

**Four** unknown operations are then executed on the min-heap. An operation is either a **removeMin** or an **insert**. The resulting state of the min-heap is shown below.

Array: [A, E, B, D, X, F, G]

(a) Determine the operations executed and their appropriate order. The first operation has already been filled in for you!

1. removeMin()
2. ________________________________
3. ________________________________
4. ________________________________

(b) Fill in the following comparisons with either >, <, or ? if unknown. We recommend considering which elements were compared to reach the final array.

1. X ____ D
2. X ____ C
3. B ____ C
4. G ____ X
Assume that the `hashCode` of a `TA` object returns `charisma`, and the `equals` method returns `true` if and only if two `TA` objects have the same first letter in their name.

Assume that the `ECHashMap` is a `HashMap` implemented with external chaining as depicted in lecture. The `ECHashMap` instance begins at size 4 and, for simplicity, does not resize. Draw the contents of `map` after the executing the insertions below:
4 Buggy Hash

The following classes may contain a bug in one of its methods. Identify those errors and briefly explain why they are incorrect and in which situations would the bug cause problems.

(a)  class Timezone {
    String timeZone; // "PST", "EST" etc.
    boolean dayLight;
    String location;
    ...
    public int currentTime() {
      // return the current time in that time zone
    }
    public int hashCode() {
      return currentTime();
    }
    public boolean equals(Object o) {
      Timezone tz = (Timezone) o;
      return tz.timeZone.equals(timeZone);
    }
  }

(b)  class Course {
    int courseCode;
    int yearOffered;
    String[] staff;
    ...
    public int hashCode() {
      return yearOffered + courseCode;
    }
    public boolean equals(Object o) {
      Course c = (Course) o;
      return c.courseCode == courseCode;
    }
  }