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 ____________ and a worst case runtime of ____________.
2. `insert` has a best case runtime of ____________ and a worst case runtime of ____________.
3. A ____________ or ____________ 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 ____________ places in the heap.
5. Given a min-heap with $2^n - 1$ elements, for an element

   - to be on the second level it must be less than ____________ element(s) and greater than ____________ element(s).
   - to be on the bottommost level it must be less than ____________ element(s) and greater than ____________ element(s).
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. Note that this question does not assume a specific ordering of operations from the previous part, i.e. we don’t know which of the two possible

1. X _____ D
2. X _____ C
3. B _____ C
4. G _____ X
3 Graph Conceptuals

Answer the following questions as either True or False and provide a brief explanation:

1. If a graph with $n$ vertices has $n - 1$ edges, it must be a tree.

2. The adjacency matrix representation is typically better than the adjacency list representation when the graph is very connected.

3. Every edge is looked at exactly twice in every iteration of DFS on a connected, undirected graph.

4. In BFS, let $d(v)$ be the minimum number of edges between a vertex $v$ and the start vertex. For any two vertices $u, v$ in the fringe, $|d(u) - d(v)|$ is always less than 2.
4 Cycle Detection

Given an undirected graph, provide an algorithm that returns true if a cycle exists in the graph, and false otherwise. Also, provide a $\Theta$ bound for the worst case runtime of your algorithm. You may use either an adjacency list or an adjacency matrix to represent your graph. (We are looking for an answer in plain English, not code).