**BinomialHeap**

**1. Test Cases for insert Function**

**Test Case 1.1: Insert into an Empty Heap**

* **Input: Insert 10 into an empty binomial heap.**
* **Expected Output:**
  + **Heap contains one tree of degree 0 with root 10.**
  + **The minimum value is 10.**

**Test Case 1.2: Insert into a Non-Empty Heap**

* **Input: Insert 20 into a heap containing {10}.**
* **Expected Output:**
  + **Two trees of degree 0: one with root 10 and one with root 20.**
  + **The minimum value remains 10.**

**Test Case 1.3: Insert Multiple Values**

* **Input: Insert values 15, 5, 25 sequentially into the heap.**
* **Expected Output:**
  + **The heap reorganizes into valid binomial trees:**
    - **Tree of degree 0: Root 25.**
    - **Tree of degree 1: Root 10 (child 15).**
    - **Tree of degree 2: Root 5 (child 10 with subtree).**
  + **Minimum value is 5.**

**2. Test Cases for findMin Function**

**Test Case 2.1: Find Minimum in Single Tree**

* **Input: A heap with a single tree: {10}.**
* **Expected Output: Minimum value is 10.**

**Test Case 2.2: Find Minimum Across Multiple Trees**

* **Input: A heap with values {25, 10, 5, 15}.**
* **Expected Output: Minimum value is 5.**

**Test Case 2.3: Empty Heap**

* **Input: Call findMin on an empty heap.**
* **Expected Output: Return null or indicate the heap is empty.**

**3. Test Cases for merge Function**

**Test Case 3.1: Merge Two Heaps of Equal Sizes**

* **Input:**
  + **Heap 1: {10, 20}.**
  + **Heap 2: {5, 15}.**
* **Expected Output: A valid heap containing {5, 10, 15, 20}, reorganized into binomial trees:**
  + **Degree 0: Root 15.**
  + **Degree 1: Root 5 (child 10 with subtree).**
  + **Minimum value is 5.**

**Test Case 3.2: Merge an Empty Heap**

* **Input: Merge a non-empty heap {10, 20} with an empty heap.**
* **Expected Output: The heap remains {10, 20}.**

**Test Case 3.3: Merge Heaps with Overlapping Degrees**

* **Input:**
  + **Heap 1: {10} (tree of degree 0).**
  + **Heap 2: {20} (tree of degree 0).**
* **Expected Output: Trees merge to form a new tree of degree 1 with root 10 (child 20).**

**4. Test Cases for deleteMin Function**

**Test Case 4.1: Delete Minimum from Single Tree**

* **Input: A heap with {10}.**
* **Expected Output: The heap becomes empty.**

**Test Case 4.2: Delete Minimum with Multiple Trees**

* **Input: A heap containing {5, 10, 15, 25}.**
* **Expected Output:**
  + **The minimum (5) is removed.**
  + **Remaining heap reorganizes into valid binomial trees:**
    - **Degree 0: Root 25.**
    - **Degree 1: Root 10 (child 15).**

**Test Case 4.3: Delete Minimum in an Empty Heap**

* **Input: Call deleteMin on an empty heap.**
* **Expected Output: No change, and an error or exception is thrown indicating the heap is empty.**

**5. Test Cases for decreaseKey Function**

**Test Case 5.1: Decrease Key of a Leaf Node**

* **Input: Decrease the key of a node with value 15 to 2 in the heap {10, 15, 25}.**
* **Expected Output:**
  + **The value 2 is swapped with its parent(s) until the heap property is restored.**
  + **New minimum value is 2.**

**Test Case 5.2: Decrease Key to Value Greater Than Current**

* **Input: Attempt to decrease the key of 15 to 20.**
* **Expected Output: No change, and an error or exception is thrown.**

**Test Case 5.3: Decrease Key in Empty Heap**

* **Input: Call decreaseKey on an empty heap.**
* **Expected Output: An error or exception is thrown.**

**6. Test Cases for delete Function**

**Test Case 6.1: Delete a Non-Minimum Node**

* **Input: Delete the node with value 15 in {5, 10, 15, 25}.**
* **Expected Output:**
  + **The value 15 is replaced with negative infinity and bubbled up to the root.**
  + **The root with negative infinity is deleted.**
  + **Remaining heap is {5, 10, 25}.**

**Test Case 6.2: Delete Root Node**

* **Input: Delete the root node (minimum) from {5, 10, 15}.**
* **Expected Output:**
  + **The minimum (5) is removed.**
  + **Heap reorganizes into valid binomial trees.**