These are some review questions to test your understanding of the material. Some of these questions may appear on an exam.

1  Priority Queues and Heaps

Note: pseudocode for merging leftist heaps is given on page 3.

1.1 Define “priority queue.”
1.2 Define “binary heap.”
1.3 Define “null path length” in a binary tree.
1.4 Define “leftist binary tree.”
1.5 Define “leftist heap.”
1.6 Insertion and deletion in a binary heap is in $O(\log n)$ on average. Explain why this is so.
1.7 Finding the minimum element in a binary heap (a min-heap) is in $O(1)$ worst case. Explain why this is so.
1.8 Prove that the largest element in a min binary heap is a leaf.
1.9 For a min binary heap of $n$ elements, what is the range of indices in which the maximum element will be found?
1.10 Describe, in English, an algorithm to find the largest element in a binary min-heap. What is the asymptotic worst case performance of your algorithm?
1.11 The array representing a binary heap contains the following elements in the order $2, 8, 3, 10, 16, 7, 18, 13, 15$. Show the order that results at each stage of inserting the element $4$.
1.12 The array representing a binary heap contains the following elements in the order $2, 8, 3, 10, 16, 7, 18, 13, 15$. Show the order that results at each stage of deleting the minimum element.
1.13 Describe, in English, the process for constructing a binary heap from a given set of initial values (i.e., heapify an array of elements).
1.14 Construct a binary heap using the initial values $18, 2, 13, 10, 15, 3, 7, 16, 8$. Show the heap at each stage of construction.
1.15 Given a drawing of a binary tree, state if it is a leftist tree and if it is a leftist heap. Give reasons.
1.16 Prove that any complete binary tree is a leftist tree.
1.17 Prove: for any leftist tree having $N$ vertices, the number of vertices, $R$, on the rightmost path to a non-full vertex is given by $R \leq \lceil \log(N + 1) \rceil$. 
1.18 Describe how to do \texttt{findMin} in a leftist heap.

1.19 Using the \texttt{merge} (aka \texttt{meld}) operation, describe how to do the \texttt{insert} and \texttt{deleteMin} operations for leftist heaps.

1.20 Describe a method to construct a leftist heap from a given set of values. Your code must construct the leftist heap of \( N \) elements in \( O(N) \) time (not in \( O(N \lg N) \)).

1.21 Given drawings of two leftist heaps \( H1 \) and \( H2 \), draw the leftist heap that results from the operation \texttt{merge}(H1, H2)
Merging Leftist Heaps

Here is pseudo-code for merging two leftist heaps. This is a destructive operation

```cpp
template <class Comparable>
Comparable & rootValue(LHeap<Comparable> & H)
{
    return the value stored at the root of H
}

template <class Comparable>
LHeap<Comparable> & leftChild(LHeap<Comparable> & H)
{
    return the left child of the root of H
}

template <class Comparable>
int leftLength(LHeap<Comparable> & H)
{
    return the null path length of the left child of the root
    of H
}

template <class Comparable>
LHeap<Comparable> & Merge(LHeap<Comparable> & H1,
                        LHeap<Comparable> & H2)
{
    if (H1 is empty) |
        {rootValue(H1) > rootValue(H2));
         // make sure min value in H1 is not larger than
         // min value in H2
         swap(H1, H2);
    
    if (H1 is empty) // if is, then both H1 and H2 are empty
        return H1;    // return an empty LHeap (or NULL)

    // recursive call to Merge
    replace rightChild(H1) by Merge(rightChild(H1), H2);

    // make sure H1 is still leftist
    if (leftLength(H1) < rightLength(H1);
        swap(left(H1), right(H1));

    return H1;    // return the modified H1
}
```