1. Given a list of $n$ integers, describe how to find the $\sqrt{n}$ smallest items, listed in increasing order, in $O(n)$ time. The input list is unsorted, but the output should be sorted. (Hint: $\sqrt{n}\lg\sqrt{n}$ and $\sqrt{n}\lg n$ are both $O(n)$.) [8 points]

2. Insert the following values into an initially empty B-tree of order $t = 4$:

   5, 9, 18, 37, 2, 7, 10, 15, 22, 30, 3, 16, 17, 23, 13, 4, 27, 21

   [12 points]

3. Lemma 8.4 of the text describes the time of RADIX-SORT on $n$ $b$-bit integers. Suppose we choose $r = \lceil\lg n\rceil$ and use COUNTING-SORT as the stable sort it mentions. What is the largest $b$ can be (in terms of $n$) with RADIX-SORT still running in time $O(n\log n)$? [8 points]

4. Consider these two routines, the first builds a binary search tree, and the second sorts an array $A$:

   ```java
   buildBST(array A) {
     BST T
     for i=1 to n
       T.insert(A[i])
     return T }
   BSTsort(array A) {
     BST T = buildBST(A)
     T.inorderTrav
   }
   ```

   (a) What is the worst case time of BSTsort?
   (b) What is the average case time of BSTsort?
   (c) Recall that $n$ calls to HEAPINSERT takes time $\Theta(n\log n)$, but a clever BUILDHEAP is $\Theta(n)$. Is something like that possible with buildBST? That is, is it possible that the time of buildBST be improved to $o(n\log n)$? Do not assume any restrictions on the range of input data, and do assume that access to the input is to be purely comparison-based.

   [10 points]
5. Regarding Fibonacci heaps, these are to be min-heaps (as in the text).

(a) Into an initially empty Fibonacci heap, insert the values: 9, 3, 2, 18, 11, 19, 25
(b) Remove the min from that heap.
(c) Into another Fibonacci heap, insert the values: 17, 27, 20, 12, 15, 4, 5, 8, 10, 13
(d) Remove the min from that second heap.
(e) Merge the two heaps together.
(f) Remove the min again.
(g) Look at the Fibonacci heap on the attachment. Decrease 26 to 7. Then decrease 33 to 21.

[13 points]

6. Regarding red-black trees

(a) Into an initially empty RB tree insert the values: 10, 18, 5, 7, 3, 8, 6, 9, 20, 25, 30.
(b) From the RB tree on the attachment (dotted lines mean red), delete 16.
(c) From the same RB tree (before the deletion), delete 10.

[14 points]

7. Recall the problem on homework 6 about the water jugs: we have $n$ yellow water jugs and $n$ green water jugs, all different shapes and sizes. Also, for each yellow water jug, there is a unique green water jug holding the same amount of water. A comparison here consists of pouring water from a jug of one color into a jug of another color (you have an unlimited supply of water). You can tell as a result of this comparison if one jug is smaller than the other, larger than the other, or exactly the same size.

Describe an $O(n)$ randomized algorithm to find the smallest green water jug.

[10 points]

**Total:** 75 points