This homework is due at the start of class on Friday, March 1. In all cases, remember to show your work and justify your answers. Note that there is a test coming up on Monday, March 4.

1. A hashtable has 13 locations and uses the hash function $hash(key) = key\%13$. It is a closed hash table, meaning all the keys are stored in the table itself, and it uses open addressing with linear probing. Show the table after the following sequence of keys is inserted into an initially empty table:

```
66, 54, 98, 2, 38, 45, 14, 17, 35, 26, 88, 64
```

2. (*From textbook exercise 12.1-1*) For the set $\{1, 4, 5, 10, 16, 17, 21\}$ of keys, draw five binary search trees containing exactly that set of keys, where the trees have heights 2, 3, 4, 5, and 6.

3. (*From textbook exercise 12.3-3*) We can sort a given set of n numbers by first building a binary sort tree containing these numbers (using the TREE-INSERT algorithm n times) and then printing the numbers by an inorder tree traversal. What are the worst-case and best-case running times for this sorting algorithm?

4. (*From textbook exercise 12.1-5*) Sorting n elements using a comparison sort takes $\Omega(n \log(n))$ time in the worst case. Using this fact, argue that any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \log(n))$ time in the worst case. (Hint: Suppose that you have an algorithm that can build binary search trees more quickly than that.)

5. Suppose that the keys 1, 2, 3, 4, 5, 6, 7, 8, 9 are inserted in that order into a B-Tree that has minimal degree $t = 2$. Initially, the tree is empty. Show the B-Tree that results. It would also be a good idea to show the tree at several stages along the way. (Note that every node in the tree contains either 1, 2, or 3 keys. An interior node contains 2, 3, or 4 child pointers.)

6. Suppose that a huge file is stored in secondary storage. It is much too large to fit into main memory. Describe an algorithm for sorting the file. You should try to minimize the number of page reads and writes. You can create extra files, of any size, on secondary storage if you need them. Say what you can about both the run time and the number of page accesses used by your algorithm. To make things definite, you might consider the case where the file is 1 TB (2^{40} bytes) and main memory is 8 GB (2^{33} bytes) and page size is 4 KB (2^{12} bytes). (You can assume, if you like, that no item in the file overlaps a page boundary. That is, when you read a page, you get some whole number of items.)