Searching

Hash Tables

1. Why does Java use 31 in the hashCode() for String?
   It's prime, so that when the user mods out by another number, they have no common factors (unless it's a multiple of 31). 31 is also a Mersenne prime (like 127 or 8191) which is a prime number that is one less than a power of 2. This means that the mod can be done with one shift and one subtract if the machine's multiply instruction is slow.

2. Is the following implementation of hashCode() legal?

```
public int hashCode() {
        return 17;

}
```

   Solution. Yes, but it would cause all keys to hash to the same spot, which would lead to poor performance.

3. Suppose that the keys A through G, with the hash values given below, are inserted in some order into an initially empty table of size 7 using a linear-probing table (with no resizing for this problem).

key	A	B	C	D	E	F	G
hash (M = 7)	2	0	0	4	4	4	2

Which of the following could not possibly result from inserting these keys?

a. E F G A C B D
b. C E B G F D A
c. B D F A C E G
d. C G B A D E F
e. F G B D A C E
f. G E C A D B F

Give the minimum and the maximum number of probes that could be required to build a table of size 7 with these keys, and an insertion order that justifies your answer.

Balanced Trees, Red-Black Trees

1. Given a sorted sequence of keys, describe how to construct a red-black BST that contains them in linear time.
   Solution. Since the input is already sorted, the best approach is to construct a perfectly balances BST in a top-down manner.
   - Recursively choose the middle element as the root
   - Recursively assign the middle of the left half as the left child and the middle of the right half as the right child
   - Repeat until all elements are placed

   This step is O(n) because we visit each node exactly once.

   After constructing the balanced BST, we need to assign red and black colors so that it satisfies red-black tree properties. The key idea is:

   - All leaves should be black
   - A node at even depth is black, and a node at odd depth is red

   To achieve this:

   - Color the root black
   - Alternate between red and black for levels below the root (red nodes will be the children of black nodes)

2. Suppose that you do a search in a red-black BST that terminates unsuccessfully after following 20 links from the root. Fill in the blanks below with the best (integer) bounds that you can infer from this fact about any unsuccessful search
   Must follow at least _____ links from the root
   Need follow at most _____ links from the root.
   Solution. Since an unsuccessful search follow 20 links, this means that the height of the tree is at least 20.
   - The shortest path in a red-black BST follows at least half of the longest path.
   - The black-height h_b is at least half of the total height:

     $$h_b \geq \frac{20}{2} = 10$$

   - Since the shortest possible path must also be at least this black-height, we infer: Must follow at least **10** links.

   The longest path can be at most twice the shortest path in a red-black BST.

   - Since 20 links were followed in an unsuccessful search, the worst case is: Need to follow at most **40** links.