Problem 1: You are given two arrays $A$ and $B$ containing elements from a set $U$. Create two lists $D$ and $S$ such that $S$ contain all the singles (i.e. elements that appear only in $A$ or only in $B$), and $D$ contains all the doubles (i.e. elements appearing in both $A$ and $B$). Your algorithm should run in linear time (i.e. $O(n)$ where $n = |A| + |B|$).

Solution: Hash all the elements of $A$ into a hash table of size $n$ resolving collisions using a linked list. Assuming simple uniform hashing each slot will have an average of $O(1)$ elements. Now hash the elements of $B$. If an element of $B$ hashed into an empty slot put it in $S$. If the element hashes into an occupied slot, then check if it appears in the list. If it does not, then you add it to $S$. If it does appear in the list, that means the element is also in $A$, therefore you add it to $D$ and remove it from the hash table. At the end all the elements left in the hash table are singles (since they are elements of $A$ that do not appear in $B$) and can be added to $S$. The running time is expected $O(n)$ since we hash each element once, and the search takes expected $O(1)$.

Problem 2: Assume simple uniform hashing. After hashing $n$ keys into a table of size $m$, what is the expected number of collisions (in the entire table)?

Solution: The probability that a pair of keys collide is $\frac{1}{m}$. There are $\frac{n(n-1)}{2}$ distinct pairs, so the expected number of collisions is $\frac{n(n-1)}{2m}$.

Problem 3: Assume that you are searching for a key $k$ in a binary search tree $T$, and that you find $k$ in a leaf. The search process partition the tree $T$ in three disjoint parts: $A$ the nodes to the left of the search path, $B$ the search path itself, and $C$ the nodes to the right of the search path. Prove or disprove: for any $a \in A, b \in B, c \in C$ we have that $a \leq b \leq c$.

Solution: The statement is false. Here is a counterexample. Search for 14 in the tree below. The key 11 is to the left of the root but is larger.

```
    10
   /  \
  8   12
 /     \
6     9   11  14
```