1. Provide pseudo-code descriptions for the operation insertFirst on singly linked lists, and on doubly linked lists.

   The operation insertFirst gets as input a list and a key, and adds a node with the input-key at the beginning of the input-list.

   Give the worst-case time complexity of your operations in terms of $O$. 

2. Provide pseudo-code descriptions for the operation insertBefore on singly linked lists, and on doubly linked lists.

   The operation insertBefore gets as input a list and a node and a key. It adds a node with the input-key before the input-node. We are given the input-node, which means that we do not have to search it.

   Give the worst-case time complexity of your operations in terms of $O$. 

3. Give a non-recursive procedure in $O(n)$ that reverses a singly linked list of $n$ elements. The procedure should use no more than constant storage space beyond the space needed for the list itself. (Is your procedure also in $\Theta(n)$?)

4. Give an implementation of a stack using a singly linked list. What is the top of the stack? Provide operations for push and pop in $O(1)$. 

5. Give an implementation of a max-priority queue using a(n unsorted) singly linked list. What is the worst-case time complexity for the operations for adding and deleting?

6. For the sequence

   30 20 56 75 31 19

   and the hash function $h(k) = k \mod 11$, construct the hash table using open addressing with linear probing.

7. Add the following numbers (in this order) to an initially empty hash table of length 11

   12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5

   using hash function $h(k) = (2 \cdot k + 5) \mod 11$ and solving collision using

   (a) chaining,
(b) linear probing,
(c) double hashing, with second hash function $h'(k) = 7 - (k \mod 7)$.

8. Suppose we want to define a hash function for English words which are hashed to (only) 26 slots. Why is it not a good idea to hash each word to the first letter of this word?

9. Determine the probability that all $n$ keys are hashed to the same place in a hash table of size $m$ (assuming that the hash function distributes keys evenly over all places in the hash table).

10. Come up with a real-world problem where hashing is useful.