Exercise 5.1  Open Hashing.

We consider open hashing with a hash table of size \( p \) for a prime \( p \).

a) Decide which of the following functions are useful as a hash function (and which are not), and justify your answer.
   - \( h(k) = \) Digit sum of \( k \)
   - \( h(k) = k(1 + p + p^2) \mod p \)
   - \( h(k) = \lfloor p(rk - \lfloor rk \rfloor) \rfloor, \ r \in \mathbb{R^+} \setminus \mathbb{Q} \)

b) Insert the keys 12, 19, 6, 15, 13, 2, 28, 24 in this order into a hash table of size 11. Use open addressing with the hash function \( h(k) = k \mod 11 \) and resolve the conflicts using
   (i) linear probing
   (ii) quadratic probing
   (iii) double hashing with \( h'(k) = 1 + (k \mod 9) \).

c) Which problem occurs if the key 13 is removed from the hash tables in a), and how can you resolve it? Which problems occur if many keys are removed from a hash table?

d) In this task we use the hash function \( h(k) = k \mod p \) and resolve collisions using double hashing. Let \( q \) be the largest prime smaller than \( p \), \( h'(k) \) the second hash function and \( s(j, k) \) the probing function. The complete hash function in the \( j \)-th step is \( h(k) - jh'(k) \mod p \) if \( h'(k) \) is given, and \( h(k) - s(j, k) \mod p \) if \( s(j, k) \) is given. Decide which of the following choices of \( h'(k) \) and \( s(j, k) \) are reasonable (and which are not), and justify your answer.
   - \( h'(k) = \lceil \ln(k + 1) \rceil \mod q \)
   - \( s(j, k) = k^j \mod p \)
   - \( s(j, k) = ((k \cdot j) \mod q) + 1 \)

e) Which advantage does double hashing have when you compare it to quadratic probing?
**Exercise 5.2  Binary Search Trees.**

a) Draw the resulting tree if you insert the keys 12, 19, 6, 15, 13, 2, 28, 24 in this order into an initially empty binary search tree.

b) Draw the resulting tree after you removed the key 12 from the tree in a).

c) Consider a binary tree with the root $v$, the left subtree $T_l$ and the right subtree $T_r$. There exist different possibilities to traverse the tree:
   - **Preorder** prints $v$, traverses $T_l$ and after that $T_r$.
   - **Postorder** first traverses $T_l$, after than $T_r$, and finally prints $v$.
   - **Inorder** first traverses $T_l$, prints $v$, and after that traverses $T_r$.

Give the preorder, postorder and inorder traversal of the tree in a).

d) Draw the binary search tree that generates the postorder traversal $9, 5, 12, 17, 19, 14, 30, 22, 21, 11$.

**Exercise 5.3  Amortized Analysis.**

In this exercise, we consider arrays that grow dynamically on demand (e.g., `java.util.Vector` in the Java standard library). Specifically, we assume that we insert the values one by one. If more than $n$ elements are stored, a new array of fixed length $k > n$ is created, the old contents are copied and the new element is stored. An array of length $k$ can be created in $k$ steps, and the copying of an element is done in constant time.

a) Describe how to choose $k$ so that each insert operation has amortized constant time, and hence the insertion of $n$ elements can be done in time $\Theta(n)$. Prove using amortized analysis that your choice results in constant amortized time per insert operation.

b) Now consider the situation where we allow to delete the last element from the array. We allow any mixture of such insert and delete operations. For memory reasons, it may be useful to also shrink the array sometimes. Describe how would you shrink the array, and show that both insertion and removal require amortized constant time.

**Hand-in:** until Wednesday, 26th March 2014.