1. Who is last?
Let’s consider the following game. A set of people have placed themselves in a circle. The persons are counted clockwise and every second person has to leave the circle. The first to leave is number 2. Finally there is only one person left, who is the winner. For instance, if you have 5 persons, what happened is the following: First ones to leave are 2 and 4. Now we are left with 1, 3 and 5. The next to leave is 1, then 3 remains and 5 leaves. So number 3 is the winner. Write a program (in pseudocode or Java) to solve the game when the number of players $n$ is given as input.

2. (a) Show how insert works in a binary search tree, when the following keys are added to an empty tree, in the following order: 2, 0, 3, 7, 9, 1, 5, 6 and 8. Draw the tree after each insertion.
(b) Simulate the delete-operation for the tree resulting from point (a). Delete the nodes with the keys 3, 9, 2 and 5, in this order. Draw the tree after each deletion.

3. (a) Draw smallest and largest AVL-trees, which have height of 4. You may choose the values of the nodes freely as long as they meet the binary search tree property.
(b) Suppose we allocate 8 bits for the height counters in each node of an AVL tree. What is the smallest number of nodes in the tree that might cause the height of the root to overflow? (It is sufficient to give an estimate for the order of magnitude.)

4. (a) Show how AVL insert works, when the following keys are added to an empty tree, in the following order: 41, 38, 31, 12, 19, 8, 27 and 49. Show how rotations are done in each insert.
(b) Do as above, but the keys are inserted to an empty tree in reverse order, 49, 27, 8, 19, 12, 31, 38 and 41.

5. AVL-delete works as follows: First a node is deleted with normal binary search tree delete operation. This may cause an unbalance to the tree. If there is unbalance, the problems are in path from the deleted node to the root of the tree. So, to correct the problems, each node in that path (starting from the parent of the deleted node) are checked and if an unbalance is found, the necessary rotations are made. In AVL-insert, a single rotate or a double rotate is enough to return the balance. With AVL-delete this is not the case. The algorithm must check each node form the path to the root node of the tree. See eg. www.cs.uga.edu/~eileen/2720/Notes/AVLTrees-II.ppt
(a) Show how AVL-delete works when it removes keys 12, 49, 31 and 8 from the tree that results in 4 (a).
(b) Remove the keys 12, 8, 49 and 41 (in this order) from the tree that results in 4 (b).
6. (a) The time requirement of the binary search tree operations $\text{min}$ and $\text{max}$ (finding the smallest and the largest node) is $O(h)$, where $h$ is the height of the tree. Change the tree implementation so, that the time requirement of operations $\text{min}$ and $\text{max}$ is only $O(1)$ and the time requirement of other operations stays the same.

(b) The time requirement of the binary search tree operations $\text{succ}$ and $\text{pred}$ is $O(h)$, where $h$ is the height of the tree. Change the tree implementation so, that the time requirement of operations $\text{succ}$ and $\text{pred}$ is only $O(1)$ and the time requirement of other operations stays the same.

(c) What are the benefits and the disadvantages of the previous changes?

7. Show that any binary search tree can be transformed into any other binary search tree (for the same keys) just by performing rotations. Estimate the number of rotation needed in the worst case. (Here the trees are of course not assumed to be balanced.)  
**Hint:** first show that any tree can be transformed into a right-going chain, i.e. a tree where no node has a left child.

8. You win the first prize in a television quiz show. The prize is to choose from $n$ electric appliances as many as you can carry. You can carry at most $k$ kg. Each appliance has a value $a_i$ euro and a weight $p_i$ kg, $i = 1, 2, \ldots, n$. Give in pseudocode an algorithm that maximizes the value of your prize.

   **Hint:** Branch-and-bound.

9. **Rotation-game:**
   - The game is played on a $3 \times 3$-table, which includes the numbers $1 - 9$.
   - At the start the numbers are on the table at random order, e.g.
     
     \[
     \begin{array}{ccc}
     5 & 7 & 1 \\
     9 & 3 & 6 \\
     8 & 2 & 4 \\
     \end{array}
     \]
   - The numbers should be put in increasing order:
     
     \[
     \begin{array}{ccc}
     1 & 2 & 3 \\
     4 & 5 & 6 \\
     7 & 8 & 9 \\
     \end{array}
     \]
   - The game is played by making rotations, e.g. rotating the right lower corner of $2 \times 2$ numbers
     
     \[
     \begin{array}{ccc}
     5 & 7 & 1 \\
     9 & 3 & 6 \\
     8 & 2 & 4 \\
     \end{array}
     \rightarrow
     \begin{array}{ccc}
     5 & 7 & 1 \\
     9 & 2 & 3 \\
     8 & 4 & 6 \\
     \end{array}
     \]
   - A rotation moves 4 adjacent numbers one step clockwise
   - A rotation can be made to any 4 adjacent squares
   - A rotation is only way to move the numbers
Give an algorithm that finds a solution to a rotation game. Give the time and space complexity for your algorithm.

Is there always a solution? Does your algorithm find a solution using the minimum number of rotations. If not, how should the algorithm be changed so that the minimum number of rotations is found?

10. Assume that you have an originally empty hashing table of size \( m = 11 \) and a hash function \( h(k) = k \mod m \). We insert keys 10, 22, 31, 4, 15, 28, 17 and 88 in this order.

(a) Draw and explain what happens, when we use chaining.

(b) Draw and explain what happens, when we use open addressing with linear probing and \( h' = h \).

(c) As before, but we use open addressing with quadratic probing and \( h' = h \ c_1 = 1 \) and \( c_2 = 3 \).

(d) As before, but we use open addressing with double hashing with \( h'(k) = k \mod m \) and \( h''(k) = 1 + (k \mod (m - 1)) \)

11. The aim is to make a hash table of the university’s students using as key the student number. There are about 35000 students at University of Helsinki.

The collisions are handled using chaining, with an expected length of the chain between 2.0 and 3.0. Suggest a suitable size \( m \) when we use the division method.

12. A 2-3-tree is a type of balanced search tree.

A tee is 2-3-tree if the following are true:

- Every non leaf node has 2 or 3 children
- All the leaf nodes have same depth

(a) Draw the greatest and smallest 2-3-tree with height 1, 2 and 3

(b) Assume that 2-3-tree has height \( h \). Show what is the smaller and upper limit of the number of the nodes.

(c) Assume that 2-3-tree has \( n \) nodes. Show what is the smaller and upper limit of the height of the tree.

The following formula might be useful: \( \sum_{i=0}^{k} a^k = \frac{a^{k+1} - 1}{a - 1} \) where \( a \neq 1 \).