1. Let $T$ be a tree. Suppose that each leaf $x$ has a positive integer value stored in the field $x.val$. For each non-leaf node $v$, let $v$.max_leaf denote the descendant leaf of maximum value (if there is a tie, $v$.max_leaf is some leaf of maximum value). Let $v$.best_child be the child of $v$ which is on the path from $v$ to $v$.max_leaf. First, for each node $v$, compute $v$.best_child, and store it in the field $v$.best_child. Next, give a recursive procedure which outputs the sequence of vertices forming the path from the root of $T$ to $T$.max_leaf; it will use the best_child values. This second phase needs to run in time proportional to the number of nodes on this path.

2. Let $T$ be a tree. Suppose there is a record for each edge in the tree. If $(v, w)$ is an edge, where $w$ is a child of $v$, then the record is reached via the field $w.prnt_edge$ in the record for node $w$. Suppose there is a value associated with each edge $(v, w)$, stored in the field $w.prnt_edge.val$. In addition, suppose that each leaf $x$ has an integer field $x.index$. Consider the path from the root of $T$ to leaf $x$, and imagine numbering the edges along this path starting at the root, namely the first edge, the second edge, and so on. The task, for each leaf $x$, is to store in the field $x.anc_val$ the value for the $x.index$-th edge on the path from the root to the leaf, if $x.index$ is no larger than the length of this path, and to store the value nil otherwise. It will be helpful to have a global array PathVal[1 : $k$] which stores the values on the current path, where $h$ is the height of tree $T$ (you may assume that $k$ is provided as part of the input). You will want to use a recursive procedure LeafVal($v, i, PathVal$), and $i$ is the depth of node $v$ (the number of edges on the path from the root to $v$).

3. Write a recursive procedure to print all $m$-digit binary numbers in decreasing order.