Homework 10 – Sample Solution

1. (15 points)
   Let S be a set defined as follows:

   Base case: 4 is an element of S
   Recursive case: If x is in S, then x^2 is in S.

   Use structural induction to prove that all element in S are even, where an even number is defined as a number that can be written in the form 2·k for an integer number k.

Proof

Base case:
The base case of the definition of S states that 4 is an element of S. Since 2 = 2·2 holds, 4 is an even number. (Note the definition of S only specifies that 4 is in S. Thus you only need to show that 4 is even in the base case. You do not need to show that 4^2 is even, for example.)

Inductive case:
Assume x is in S and further assume that x is even. The recursive case of the definition of S says that x^2 is in S as well. We are going to show that x^2 is even as well:
Since x is even, we can write x as 2·k for an integer number k. Then $x^2 = (2\cdot k)^2 = 2\cdot(2\cdot k^2)$. Since $2\cdot k^2$ is an integer number if k is an integer number, x is even. □

2. (15 points)
   We define a perfect ternary tree as follows:

   Base case: A single node is a perfect ternary tree.
   Recursive case: If T_1, T_2, and T_3 are perfect ternary trees with roots r_1, r_2, and r_3 respectively, and T_1, T_2, and T_3 have distinct nodes and T_1, T_2, and T_3 have the same height, then the structure with root r and with an edge from r to root r_1, an edge from r to r_2, and an edge from r to r_3 is a perfect ternary tree.

   Use structural induction to prove that a perfect ternary tree has $(3^{h+1} - 1) / 2$ nodes where h is the height of the tree.

Proof

Base case:
The base case of the definition of ternary trees defines a single node to be a ternary tree. In case of a single node the tree height h is 0 and obviously n = 1. Then the following equation holds:

$$(3^{h+1} - 1) / 2 = (3^{0+1} - 1) / 2 = (3 - 1) / 2 = 1 = n.$$

Induction step:
Let T be the tree constructed as specified in the recursive case of the definition of ternary trees. That is, T is a tree with a root r and the three subtrees T_1,T_2,T_3 of the root r. Each subtree is ternary tree with the same height h. We assume that the claim holds for each of

the three subtrees. Thus if n_1, n_2, and n_3 are the number of nodes of T_1,T_2, and T_3 , respectively, then the relationships between the number of nodes and heights hold:

$$n_i = (3^{h+1} - 1) / 2 \text{ for } i = 1, 2, 3 .$$

We need to show that the number of nodes of T is $(3^{h_T+1} - 1) / 2$ where h_T is the height of T. The nodes of T are the nodes of T_1,T_2, and T_3 and the root node of T. Therefore the number of nodes of T is

$$\text{number nodes of } T = n_1 + n_2 + n_3 + 1$$

$$= (3^{h+1} - 1) / 2 + (3^{h+1} - 1) / 2 + (3^{h+1} - 1) / 2 + 1$$

$$= (3^{h+1} - 1 + 3^{h+1} - 1 + 3^{h+1} - 1) / 2 + 1$$

$$= (3 \cdot 3^{h+1} - 3) / 2 + 1$$

$$= (3^{h+1+1} - 3) / 2 + 1$$

$$= (3^{h_T+1} - 3) / 2 + 1 \quad (\text{since } h_T = h + 1)$$

$$= (3^{h_T+1} - 3) / 2 + 2 / 2$$

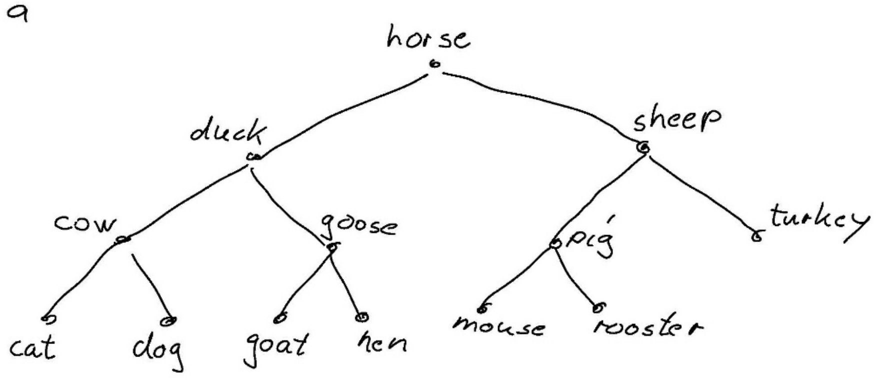
$$= (3^{h_T+1} - 3 + 2) / 2$$

$$= (3^{h_T+1} - 1) / 2 .$$

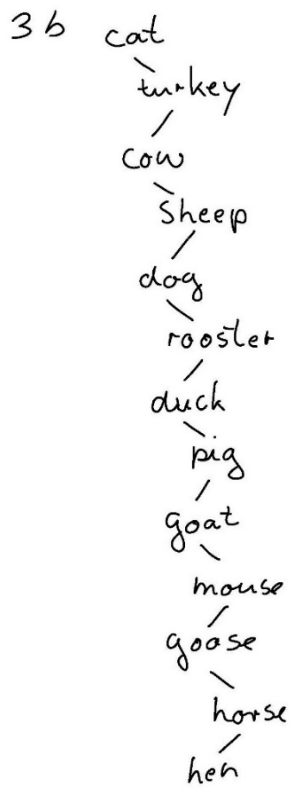
□

3. (12 points)
   Draw a binary search tree with the nodes `cat`, `cow`, `dog`, `duck`, `goat`, `goose`, `hen`, `horse`, `mouse`, `pig`, `rooster`, `sheep`, `turkey` and that meets the following requirement:
   a) The tree is complete.
      The solution tree is unique, that is, the only possible complete binary search tree with the specified values is as follows:

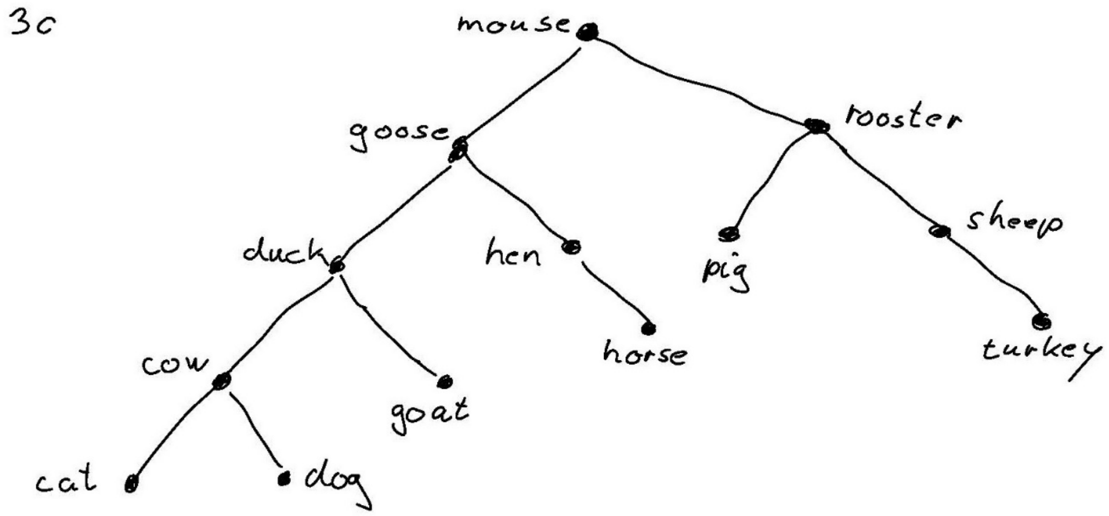
3a



b) The tree has the greatest possible height.
There are many possible solutions. Here is one possibility:



c) The tree is height-balanced and has the greatest possible height among the height-balanced trees with the given nodes.
The greatest possible height of a height-balanced binary search tree (which is an AVL tree) wit h13 nodes is 4. There may different possible tree structures. One possible tree is as follows:



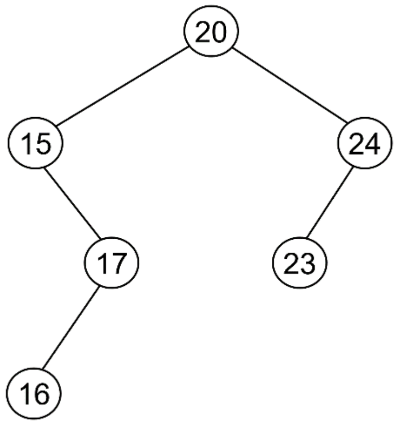
4. (10 points)
Determine the balance factor of the specified nodes and decide whether the tree is an AVL tree. The numbers represent the comparable node values.
The balance factor at a node is the height of its right subtree minus the height of its left subtree. It is NOT the balance factor of its right child minus the balance factor of its left child.

a) Balance factor of 20: **-1**

Balance factor of 15: **2**

Balance factor of 17: -**1**

Is the tree an AVL tree? **No, the tree is not height-balanced. The balance factor at 15 is greater than 1, which means that its right subtree has a height that differs by more than from the height of its left subtree.**

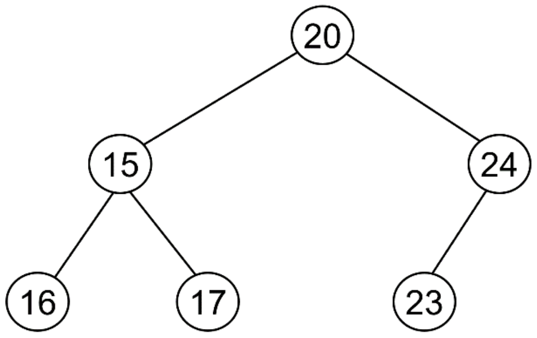


b) Balance factor of 20: **0**

Balance factor of 15: **0**

Balance factor of 16: **0**

Is the tree an AVL tree? No. The tree is height-balanced, but it does not have the binary search tree property. Node 15 has a left child greater than 15.


20
15
24
16
17
23