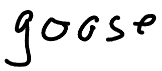
HW 9 Part B Exercise 1

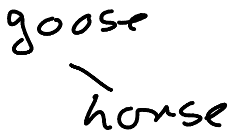
Wednesday, April 8, 2020 10:46 AM

(1) `insert("goose")`

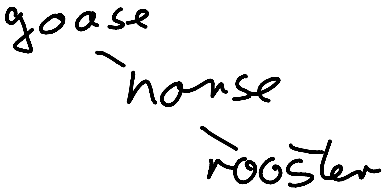
The resulting tree consist of the single node `goose`:



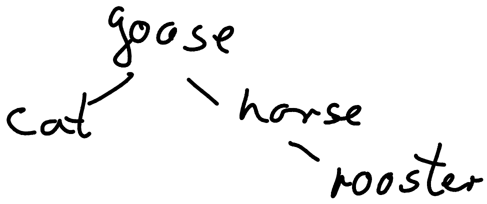
(2) `insert("horse")`



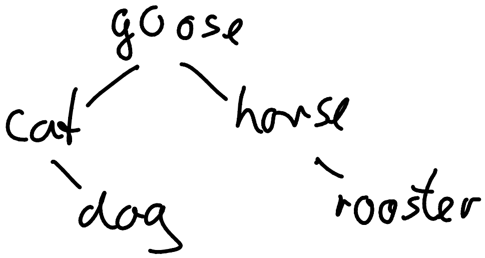
(3) `insert("rooster")`



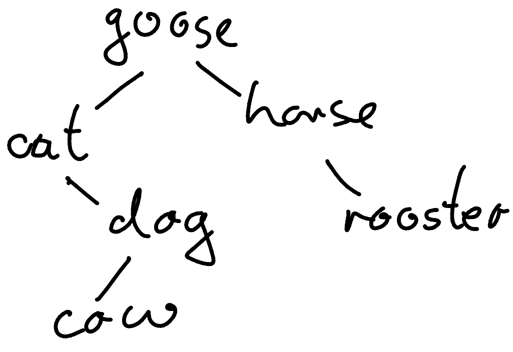
(4) `insert("cat")`



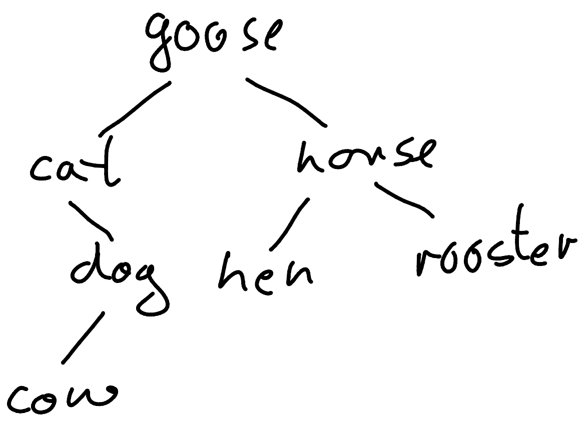
(5) `insert("dog")`



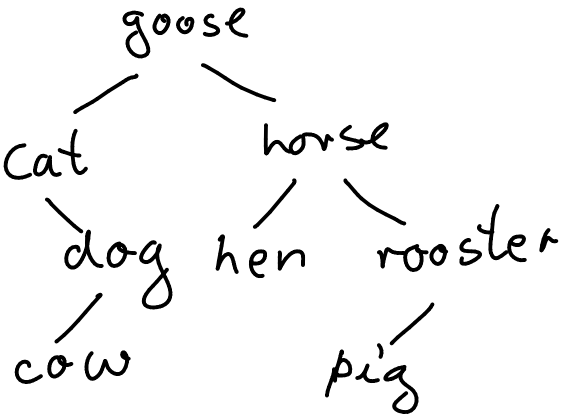
(6) `insert("cow")`



(7) `insert("hen")`

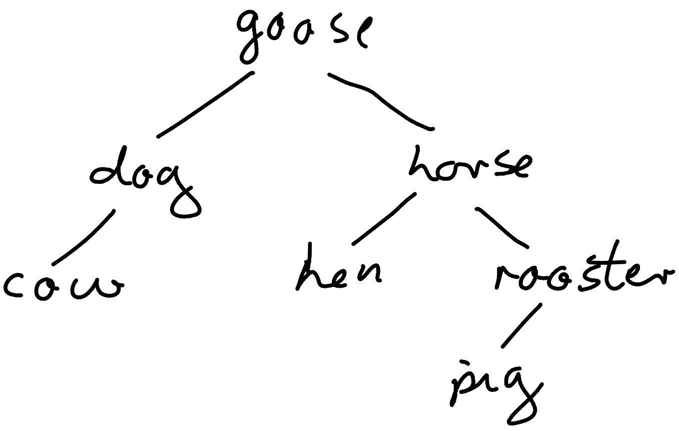


(8) `insert("pig")`



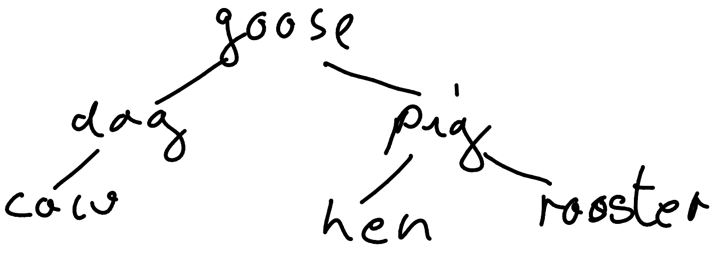
(9) `delete("cat")`

`cat` has only one child and therefore is replaced by its only child:



(10) `delete("horse")`

`horse` has two children and therefore is replaced by its successor:



Exercise 2

```
/**
 * BinaryTree
 *
 * @author CS3151
 * @param <T> type of the node values
 */
public class BinaryTree<T> {
    private BinaryNode root;

    /**
     * Instantiates a new binary tree with three nodes: The root of the new tree has
     * two children where the root has the specified value valueRoot, the left child
     * of root has the specified value valueLeft, and the right child of root has
     * the specified value valueRight.
     *
     * @precondition valueRoot != null && valueLeft != null && valueRight != null
     * @param valueRoot  value of the root
     * @param valueLeft  value of the root's left child
     * @param valueRight value of the root's right child
     */
    public BinaryTree(T valueRoot, T valueLeft, T valueRight) {
      if (valueRoot == null) {
         throw new IllegalArgumentException("the value of the root cannot be null");
      }
      if (valueLeft == null) {
         throw new IllegalArgumentException("the value of the root's left child cannot be null");
      }
      if (valueRight == null) {
         throw new IllegalArgumentException("the value of the root's right child cannot be null");
      }
      this.root = new BinaryNode(valueRoot);
      this.root.left = new BinaryNode(valueLeft);
      this.root.right = new BinaryNode(valueRight);
      this.root.left.parent = this.root;
      this.root.right.parent = this.root;
    }

    /**
     * Adds a new node with the specified value as a left child of the specified
     * node. If parentNode has already a left child, then the left child of
     * parentNode becomes the left child of the new node.
     *
     * @precondition node != null && value != null
     * @param value      the value of the new node to be added
     * @param parentNode the parent of the new node
     */
    public void addAsLeftChildOf(T value, BinaryNode parentNode) {
      if (parentNode == null) {
         throw new IllegalArgumentException("node cannot be null");
      }
      if (value == null) {
         throw new IllegalArgumentException("value cannot be null");
      }
```

```
       BinaryNode newNode = new BinaryNode(value);
       newNode.parent = parentNode;
       newNode.left = parentNode.left;
       parentNode.left = newNode;
       if (newNode.left != null) {
          newNode.left.parent = newNode;
       }
    }

    …

    /**
     * Class BinaryNode
     *
     * @author CS3151
     */
    protected final class BinaryNode {
        private T value;
        private BinaryNode parent;
        private BinaryNode left;
        private BinaryNode right;

        private BinaryNode(T value) {
            this.value = value;
            this.parent = null;
            this.left = null;
            this.right = null;
        }
    }
}
```