**General Instructions**:

1. **Submission date: 27.10.2016**, one day before moed aleph.
2. Write the solutions to all the questions in a single \*.pl file.

It should include the knowledgebase and predicates for all the questions.

1. Write your name and ID in a comment at the top of the file.
2. Make sure your code runs smoothly with SWI-Prolog 6.6.x or later.

(I have the 7.2 version)

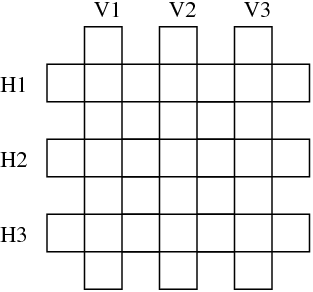
1. You can submit in singles, pairs, or triplets. If you submit in pairs or triplets, since we use the Moodle system, then each student should submit the files by himself, and all students submitting together should include all their names and IDs in the comment at the top of the files.
2. Use the exact predicate names and arity defined in the exercise. You can define and use helping predicates for your solutions.
3. **Write readable code**: use meaningful names for predicates and variables, don't overcomplicate things. Working code that is overcomplicated will lose (some) points.

**Question 1 (30 Pts)**:

Here are six Italian words: astante, astoria, baratto, cobalto, pistol, statale. The following knowledge base represents a lexicon containing these words:

word(astante, a, s, t, a, n, t, e).  
word(astoria, a, s, t, o, r, i, a).  
word(baratto, b, a, r, a, t, t, o).  
word(cobalto, c, o, b, a, l, t, o).  
word(pistola, p, i, s, t, o, l, a).  
word(statale, s, t, a, t, a, l, e).

The words are to be arranged, crossword puzzle fashion, in the following grid:



Write a predicate crossword/6 that tells us how to fill in the grid.

The first three arguments should be the **vertical** words from left to right, and the last three arguments should be the **horizontal** words from top to bottom, i.e. crossword(V1,V2,V3,H1,H2,H3).

**Important:**

* **Make sure you define the restrictions for letters that have to be the same. For example, the second letter in V1 and the second letter in H1 have to be the same letter.**
* **Make sure that each word is placed only once in the crossword.**

**Question 2 (30 Pts):**

Binary trees are trees where all internal nodes have exactly two children.

The smallest binary trees consist of only one leaf node.

We will represent leaf nodes as leaf(Label).

For instance, leaf(3) and leaf(7) are leaf nodes, and therefore small binary trees. Given two binary trees T1 and T2 we can combine them into one binary tree using the functor tree/2 as follows: tree(T1,T2) .

So, from the leaves *leaf(1)* and *leaf(2)* we can build the binary tree: *tree(leaf(1),leaf(2))*.

And from the trees *tree(leaf(1),leaf(2))* and *leaf(4)* we can build the binary tree: *tree(tree(leaf(1),leaf(2)),leaf(4))*.

Now, define a predicate mirror/2, which produces the mirror image of the binary tree that is its first argument.

For example:

   ?- mirror(tree(tree(leaf(1), leaf(2)), leaf(4)), T).  
   T = tree(leaf(4), tree(leaf(2), leaf(1))).

The figure below demonstrates how the first tree is mirrored to the second tree.

**Question 3 (40 Pts):**

Write a predicate swapFirstLast/2 which gets two lists and checks whether the first list is identical to the second, except that the first and last elements are exchanged. For example, swapFirstLast([1,2,3,4],[4,2,3,1]) should return true.

Do this in two ways (20 Pts each):

1. Define swapFirstLast/2 using recursion **without the use of any other predicate.**
2. Define swapFirstLastWithAppend/2 by using the append/3 predicate we've **learned in class** (LPNChapter6).