TP 5

1. Define two predicates, *evenlength/1* and *oddlength/1*, which take a list as their argument, and succeed if that list has an even length and and odd length respectively.

   Ex:

   ```
   | ?- evenlength(1, 2, 3, 4])
   yes
   | ?- oddlength([1])
   yes
   ```

2. Define a predicates *flatten/2*, whose first argument is a list of any complexity, and which succeeds by instantiating its second argument to a 'flattened' version of the list which contains of all the embedded list.

   Ex:

   ```
   | ?- flatten([a, [[b]], [[], [d, e]]], X).
   X = [a, b, d, e]
   yes
   ```

3. Define a predicates *count_atoms/2*, which accepts a list containing atoms, numbers and/or such lists, and 'returns' a list containing terms of the form *Item=Count* to show how many time an Item appears in any embedded list.

   Ex:

   ```
   | ?- count_atoms([a, b, b, [[a], [c], b]], L).
   L = [a=2, b=3, c=1]
   yes.
   ```

   Note that the order of the count terms in the result doesn't matter; any order will do.

4. Define a predicate *replace_elements/4*, which replaces all occurrences of a given element in a list by another, and instantiates a given variable to the answer. The arguments should be, in order:
   1. the element to be replaced;
   2. the element to replace it with;
   3. the list to do the replacing in;
   4. a variable to be instantiated to the final list.

   (The predicate needn't bother to delve inside lists within lists.)

   Ex:

   ```
   |? replace_elements(pronoun(he), pronoun(we), [pronoun(he), verb(said),
   pronoun(he), verb(did)] , L).
   L = [pronoun(we), verb(said), pronoun(we), verb(did)]
   yes.
   ```

5. Define a predicate *unifiable/3 (List, Term, Answer)*, where *Answer* is to be instantiated to all those therms in *List* which could unify with *Term*. Make sure that they are not actually unified, though.

   Ex:

   ```
   | ?- unifiable([X, a, t(Y)], t(a), L).
   ```

   should give the answer:

   ```
   L = [X, t(y)]
   ```

   and not:

   ```
   L=[t(a), t(a)]
   ```

   (Hint: consider the behaviors of *\+(Element=Term)* carefully).