Set of exercises 2

Exercise 1

Write a Prolog program that can model the following world:

- The antelope Emma is a herbivorous animal.
- The lion Harry is a ferocious animal.
- A ferocious animal is a carnivorous.
- A carnivorous animal eats meat.
- A herbivorous animal eats grass.
- All animals drink water.
- Any carnivorous can eat any herbivorous.
- Any animal consumes what it drinks or eats.

Write ONE Prolog goal that can answer these questions : Is there a ferocious animal in this world, and what does it consume?

Exercise 2

Suppose we have the following facts:

```
animal(lion).
animal(monkey).
animal(alligator).
animal(elephant).
animal(bison).
animal(gecko).
animal(swan).
animal(antelope).
animal(ant).
animal(anaconda).
animal(onager).
animal(tortoise).
animal(yak).
animal(koala).
```

Write a predicate **`mutant/1`** which generates all the possible names of animals built considering the name of an animal A1 and concatenating a second name of animal A2 having a common prefix with a suffix of A1. For example, given the facts above, we must get this:

```
?- mutant(A).
A = lionager ;
A = monkeyak ;
A = alligatortoise ;
A = elephantelope ;
A = elephant ;
A = elephantortoise ;
A = bisonager ;
A = geckonager ;
A = geckoala ;
A = swantelope ;
A = swant ;
A = swanaconda ;
A = antelopelephant ;
A = antelope ;
A = antortoise ;
A = anacondalligator ;
A = anacondantelope ;
A = anacondant ;
A = anacondanaconda ;
A = tortoiselephant ;
A = yakoala ;
A = koalalligator ;
A = koalantelope ;
A = koalant ;
A = koalanaconda ;
false
```

To write the predicate mutant/1 you will need the built-in predicate **`name/2`** defined by: name(A,L) is true if L is the list of the ASCII codes of the atom A. For example:

```
?- name(asterix,L).
L = [97, 115, 116, 101, 114, 105, 120].

?- name(Atom,[97, 115, 116, 101, 114, 105, 120]).
Atom = asterix
```

Exercise 3

Define the following predicates that specify some relationships between lists.

- **`is_in/2`**. is_in(X,L) is true if X is an element of the list L. For example:

```
?- is_in(1,[3,4,1,6]).
true

?- is_in(1,[3,4,7,6]).
false

?- is_in(X,[3,4,7,6]).
X = 3 ;
X = 4 ;
X = 7 ;
X = 6 ;
```

- **first/2**. first(X,L) is true if X is the first element of the list L. For example:

```
?- first(3,[3,7,9]).
true.

?- first(1,[3,7,9]).
false.

?- first(X,[3,7,9]).
X = 3.
```

- **mylast/2**. mylast(X,L) is true if X is the last element of the list L. For example:

```
?- mylast(9,[3,7,9]).
true.

?- mylast(1,[3,7,9]).
false.

?- mylast(X,[3,7,9]).
X = 9.
```

- **element_k/2**. element_k(X,L,K) is true if the value X is in position K in the list L. For example:

```
?- element_k(3,[8,6,10,15,4,7],4).
false.

?- element_k(15,[8,6,10,15,4,7],4).
true.

?- element_k(X,[8,6,10,15,4,7],4).
X = 15
```

- **myreverse/2**. myreverse(L1,L2) is true if L2 is the list L1 reversed. For example:

```
?- myreverse([5,4,3,2,1],[1,2,3,4,5]).
true.

?- myreverse([1,2,3,4,5],L).
L = [5, 4, 3, 2, 1].

?- myreverse(L,[1,2,3,4,5]).
L = [5, 4, 3, 2, 1] ;
```

  Hint: You will write a first simple version using the built-in predicate append/3.
  Then you will try to imagine a more efficient version that doesn't use the built-in predicate append/3.

- **is_palindrome/1**. is_palindrome(L) is true if the list L is a palindrome.

```
?- is_palindrome([x,y,z,y,x]).
true.

?- is_palindrome([x,y,z,y,x,t]).
false.
```

- **duplicate/2**. duplicate(L1,L2) is true if the elements of L1 are duplicated in the list L2.

```
?- duplicate([x,y,y,z],L).
L = [x, x, y, y, y, y, z, z].

?- duplicate(L,[x, x, y, y, y, y, z, z]).
L = [x, y, y, z]
```

- **duplicate/3**. duplicate(L1,N,L2) is true if the elements of L1 are duplicated N times in the list L2.

```
?- duplicate([x,y,y,z],3,L).
L = [x, x, x, y, y, y, y, y, y, z, z, z].

?- duplicate(L,3,[x,x,x,y,y,y,y,y,y,z,z,z]).
L = [x, y, y, z] ;
```

- **compress/2**. compress(L1,L2) is true if L2 is equal to L1 without any consecutive duplicated value. For example:

```
?- compress([a,a,a,a,b,b,c,d,d,d,e,f,f],L).
L = [a, b, c, d, e, f]
```

- **split/4**. split(L,N,L1,L2) is true if L is splitted into two sublists L1 and L2, L1 containing the first N values of L. For example:

```
?- split([a,b,c,d,e,f,g,h,i,j,k],3,L1,L2).
L1 = [a, b, c],
L2 = [d, e, f, g, h, i, j, k]
```

```
?- split([a,b,c,d,e,f,g,h,i,j,k],25,L1,L2).
false.
```

- **remove_at/4**. remove_at(V,L1,K,L2) is true if the element removed in L1 in position K is the value V and the resulting list is L2.

  ```
  ?- remove_at(V,[a,b,c,d,e,f],3,L).
  V = c,
  L = [a, b, d, e, f]

  ?- remove_at(V,[a,b,c,d,e,f],35,L).
  false.
  ```

- **insert_at/4**. insert_at(V,L1,K,L2) is true if the list L2 results from the insertion in position K of the value V in the list L1.

  ```
  ?- insert_at(c,[a, b, d, e, f],3,L).
  L = [a, b, c, d, e, f]

  ?- insert_at(c,[a, b, d, e, f],3,[a, b, x, d, e, f]).
  false.
  ```