4 Coupure

Exercice 1

On considère le programme suivant :

```
app(X,[X|_]).
app(X,[_|L]) :- app(X,L).
mp(X,L) :- app(X,L), 0 is X mod 2.
mpc(X,L) :- app(X,L), !, 0 is X mod 2.
```

1. Dressez l'arbre de recherche produit par le but `mp(X, [2,3,4])` en exhibant les substitutions produites en cas de succès.
2. Même question pour le but `mpc(X,[2,3,4])`.
3. La clause `mpc(X,L)` permet-elle toujours de calculer un membre pair X de la liste L donnée ? Quelles sont les réponses pour la requête `mpc(X,[1,2,3,4])` ?

Exercice 2

1. Ecrire un prédicat `position(+X,+L,?N)` qui réussit avec N la position de X dans la liste L. Par exemple

   ```
   ?- position(3,[1,2,3,4,3,5,3],N).
   N = 3 ;
   N = 5 ;
   N = 7
   ```

2. Ecrire un prédicat `prem_pos(+X,+L,?N)` qui réussit avec N la position de la première occurrence de X dans la liste L.

   ```
   ?- prem_pos(3,[1,2,3,4,3,5,3],N).
   N = 3
   ```

3. Ecrire un prédicat `ad(L,X)` qui définit la relation où X est l'avant dernier élément de la liste L.
4. Ecrire un prédicat `kd(K,L,X)` qui définit la relation où X est le k-ième à partir de la fin de la liste L. Conseil : vous pouvez utiliser le prédicat pré-définit `length(L,N)`.

   ```
   ?- kd(4,[a,b,c,d,e,f,f],X).
   X = d
   ```

Exercice 3

1. Ecrire un prédicat `maxliste(+L,?M)` qui réussit avec M la valeur maximale de la liste L.
2. Ecrire un prédicat `fusion(+L1,+L2,?L3)` qui fusionne deux listes ordonnées L_1 et L_2 en une seule liste ordonnée L_3. Par exemple :

   ```
   ?- fusion([1,3,4,6,10], [2,4,5,7,9],L).
   L = [1,2,3,4,4,5,6,7,9,10]
   ```