Prolog - TP1
Représentation de problèmes, Unification

Mise en route:

- lancez GNU Prolog à l'aide de la commande gprolog
- pour charger un fichier, utilisez `consult/1`. Par ex `? consult('fichier.pro').` où fichier.pro doit se trouver dans le répertoire de lancement. Alternativement utilisez `? [fichier].` (sans l'extension).
- pour lister la base de connaissances, utilisez `listing/0`.
- pour utiliser le debug, utilisez `trace/0` et `notrace/0`.
- pour arrêter, utilisez le prédicat `halt/0`.

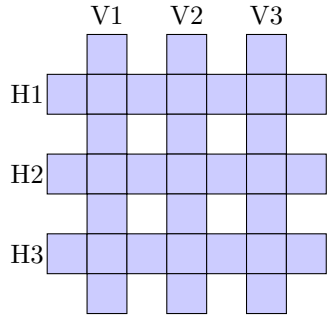
Exercice 1 Premiers pas

1. Représentez les assertions suivantes en prolog :
   a) Harry est un magicien
   b) Hagrid fait peur à Dudley
   c) Tous les magiciens sont magiques
   d) L'oncle Vernon déteste tout ce qui est magique
   e) La tante Pétunia déteste tout ce qui est magique ou fait peur à Dudley
2. Vérifiez votre programme en testant les requêtes :
   a) Estce que Pétunia déteste Hagrid ? (oui)
   b) Qui l'oncle Vernon détestet'il ? (Harry)
   c) Qui la tante Pétunia détestet'elle ? (Harry et Hagrid, utilisez le point virgule pour avoir la seconde solution).

Exercice 2 Puzzle

On cherche toutes les façons de placer dans une grille de mots croisés les mots : *abalone*, *abandon*, *anagram*, *connect*, *elegant*, *enhance*. Vous devez définir une règle de prédicat `puzzle/6` dont les trois premiers arguments sont les mots dans les colonnes V1,V2,V3, et les trois suivants sont les mots dans les lignes H1,H2,H3. Vous vous appuierez sur la base de connaissances ci-dessous. Vous pourrez éventuellement utiliser la variable anonyme (`_`) pour simplifier l'écriture de la règle[1].

[1] la variable anonyme est une variable muette qui n'est pas retournée et qu'on peut utiliser plusieurs fois dans une règle sans coréférence (plusieurs variables anonymes peuvent coexister dans une règle sans imposer la même valeur).



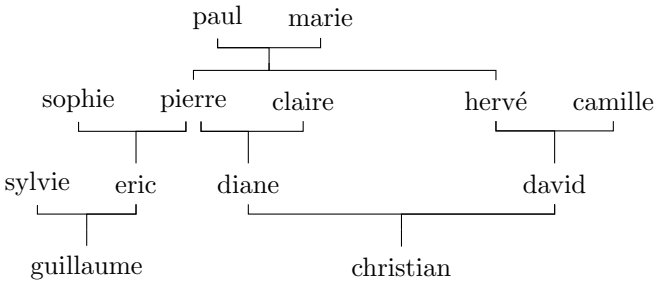

```
mot(abalone,a,b,a,l,o,n,e).
mot(abandon,a,b,a,n,d,o,n).
mot(enhance,e,n,h,a,n,c,e).
mot(anagram,a,n,a,g,r,a,m).
mot(connect,c,o,n,n,e,c,t).
mot(elegant,e,l,e,g,a,n,t).
```

Exercice 3 Négation de l'unification

On a vu en cours l'unification et le prédicat `=/2` permettant de la tester. Il peut être utile de préciser dans les règles que l'unification ne doit pas réussir, et on peut utiliser le prédicat `\=/2` vrai lorsque `=/2` est faux. C'estàdire par exemple `X \= Y` est vrai lorsque `X` ne s'unifie pas à `Y`. A l'aide de ce prédicat, écrivez les règles pour définir correctement une ligne verticale et une ligne horizontale au moyen des prédicats `verticale/1`, `horizontale/1`, `ligne/2`, et `point/2`. Par exemple `verticale(ligne(point(1,2), point(1,3)))`. Testez enfin : `verticale(ligne(point(1,2), point(1,X)))`. Qu'observezvous ? Etudiez le mode trace (`trace/0`) de cette requête et déterminez la raison du comportement observé.

Exercice 4 Généalogie

Modélisez la famille suivante :



Ecrivez ensuite les prédicats et les règles correspondant à : `homme/1`, `femme/1`, `pere/2`, `mere/2`, `parent/2`, `frere/2`, `soeur/2`, `oncle/2`, `tante/2`, `cousin/2`.

Cherchez comment il serait possible d'écrire un prédicat `ancetre/2`, vrai si le premier individu est un ancêtre du second. Ecrivez le prédicat `famille/2`, vrai si deux individus partagent un ancêtre commun. Enfin, écrivez le prédicat `consanguin/3`, vrai lorsqu'un enfant a deux parents de la même famille.

Exercice 5 Voisinage

On suppose que vous êtes un peintre en bâtiment et deviez peindre trois maisons d'une rue en sachant que la maison bleue doit être située à côté de la maison rouge, et que la maison verte doit être située à côté de la maison bleue.

Modélisez ce problème en Prolog afin de déterminer quelle maison doit être peinte de quelle couleur. Vous définirez un prédicat `couleur/3`, tel qu'il associe à chaque argument un terme dénotant la couleur. Par exemple `?couleur(X,Y,Z).` retournera `X = rouge(m1) Y = bleu(m2) Z = vert(m3)`.