Pendu

Le but de ce TME est de manipuler des *mots à trous*, c'est-à-dire des mots où certaines lettres sont inconnues, à la manière d'un jeu de pendu. Le premier exercice permet de construire un pendu en cours, en remplaçant par des trous toutes les lettres qui n'ont pas été encore proposées. On propose aussi une variante où le nombre de lettres manquantes n'est pas connu. Le second exercice, indépendant du premier, s'attache à résoudre un pendu en s'appuyant sur un dictionnaire de mots envisagés. Il s'agit de trouver tous les mots de ce dictionnaire qui pourraient être une réponse au pendu représenté par un mot à trous donné.

Note - Spécifications d'entrée/sortie des paramètres. Dans l'énoncé, lorsque l'on demande de définir un prédicat, on utilise la notation `+` ou `?` devant une variable pour préciser si la variable doit avoir une valeur ou non. Ainsi `+X` signifie que la variable est une variable d'entrée et qu'il faut lui donner une valeur lorsque l'on fait une requête, alors que `?R` signifie que la variable `R` peut être utilisée comme entrée ou comme sortie : il n'est pas nécessaire de lui donner de valeur lorsque l'on fait une requête. Il s'agit d'une spécification pour vous indiquer quel paramètre sont susceptibles de ne pas avoir de valeur et non d'une syntaxe prolog. Ainsi si on demande de définir un prédicat `f(+X,?R)`, où X et Y sont des entiers, il faut que le prédicat gère bien les requêtes telles que `f(5,R)` ou `f(5,6)`, mais il n'est pas nécessaire de considérer des requêtes comme `f(X,6)` ou `f(X,Y)`.

On rappelle de plus que dans prolog le symbole '_' peut être utilisé pour désigner une variable anonyme.

Réponses multiples. On demandera à ce que les prédicats se comportent bien vis-à-vis des demandes d'autres réponses (quand on appuie sur ';' après une réponse). Cela signifie que la requête ne devra être satisfaite que pour des valeurs correctes (par contre, on ne se soucie pas de l'éventuel false final qui signifie juste qu'il n'y a plus d'autres réponses).

Exercice 1 Génération de mots à trous

On considère ici des *mots*, comme "ouistiti", représentés par la liste de leurs lettres (`[o,u,i,s,t,i,t,i]`), ainsi que des *mots à trous*, comme "---st-t-", qui seront représentés par des listes de lettres et de variables libres (`[X1,X2,X3,s,t,X4,t,X5]` ou `[_,_,_,s,t,_,t,_]`).

1. Définir le prédicat `cacheLettre/3` tel que `cacheLettre(+M, +C,?MT)` est satisfait si `MT` est le mot à trous obtenu en remplaçant chaque occurrence de la lettre `C` par un trou (*i.e.* une variable libre) dans le mot `M`.

   Ainsi on doit avoir

   ```
   ?- cacheLettre([o,u,i,s,t,i,t,i],i,MT).      ---->    MT=[o,u,_G1,s,t,_G2,t,_G3].
   ```

2. Définir le prédicat `appartient/2` tel que `appartient(?X, +Alphabet)` est satisfait si l'élément `X` appartient à la liste de lettres `Alphabet`, une valeur devant être attribuée à `X` s'il s'agit d'une variable libre.

   Ainsi on doit avoir

   ```
   ?- appartient(a,[a,b,c,d]).                  ---->    true.
   ?- appartient(z,[a,b,c]).                    ---->    false.
   ?- appartient(X,[a,e]).                      ---->    X=a ;
                                                         X=e ;
                                                         false.
   ?- X=e, appartient(X,[a,b]).                 ---->    false.
   ```

3. Définir le prédicat `cacheGroupeLettre/3` tel que `cacheGroupeLettre(+M, +G,?MT)` est satisfait si `MT` est le mot à trous obtenu en remplaçant chaque occurrence d'une lettre appartenant à la liste de lettres `G` par un trou (*i.e.* une variable libre) dans le mot `M`.

   Ainsi on doit avoir

```
?- cacheGroupeLettre([o,u,i,s,t,i,t,i],[o,u,i],MT).
MT=[_G1,_G2,_G3,s,t,_G4,t,_G5].
?- cacheGroupeLettre([o,u,i,s,t,i,t,i],[i,t],MT).
MT=[o,u,_G1,s,G_2,_G3,_G4,_G5].
```

4. On considère maintenant des mots à trous variables, pour lequel chaque trou peut être complété par un mot (mot vide inclus). On ne doit donc plus avoir de trous consécutifs. Définir le prédicat `cacheGroupeLettreVar/3` tel que `cacheGroupeLettreVar(+M, +G,?MT)` est satisfait si `MT` est le mot à trous variables obtenu en remplaçant chaque séquence consécutive de lettres appartenant à la liste de lettres `G` par un trou (*i.e.* une variable libre) dans le mot `M`.

   Ainsi on doit avoir

```
?- cacheGroupeLettreVar([o,u,i,s,t,i,t,i],[o,u,i],MT).
MT=[_G1,s,t,_G4,t,_G5].
?- cacheGroupeLettreVar([o,u,i,s,t,i,t,i],[i,t],MT).
MT=[o,u,_G1,s,G_2].
```

   Indice : une fonction auxiliaire peut être utile pour réaliser cette fonction. La première option est de définir un prédicat auxiliaire `enlevePref(+M,+G,?MT)` qui est satisfait quand `MT` est le suffixe de `M` commençant par le premier élément de `M` qui n'appartient pas à `G` (par exemple enlevePref([o,u,i,s,t,i,t,i],[o,u,i],MT) est satisfait pour `MT=[s,t,i,t,i]`). Une autre option est de définir un prédicat auxilaire `cacheGLVAux(+M,+G,?MT,+Mode)` permettant de spécifier un Mode valant soit `insert` soit `noinsert` pour déterminer si l'on rajoute ou non une variable anonyme si le début du mot `M` est dans `G`. Ainsi `cacheGLVAux([o,u,i,s,t,i,t,i],[o,u,i],MT,insert)` est satisfait pour `MT=[_G1,s,t,_G2,t,_G3]` tandis que `cacheGLVAux([o,u,i,s,t,i,t,i],[o,u,i],MT,noinsert)` est satisfait pour `MT=[s,t,_G1,t,_G2]`.

Exercice 2 Remplissages de mots à trous

Cet exercice est indépendant du premier exercice, et peut être traité séparément. On rappelle qu'un mot est représenté par une liste de lettres, et qu'un mot à trous, comme "---st-t-", est représenté par une liste de lettres et de variables libres (correspondant aux trous).

De plus, on appelle *alphabet* un ensemble de lettres, que l'on représente par une liste de lettres (l'ordre n'ayant pas d'importance). On dit alors qu'un mot M *remplit* un mot à trous MT dans l'alphabet A si et seulement si M ne contient que des lettres de A et on peut remplacer les trous de MT par des lettres de A de sorte à obtenir M. Par exemple, "amiral" et "emirat" remplissent "-m-r–" dans l'alphabet {a,e,i,l,m,r,t}. De même, "amorce" et "amoral" remplissent ce même mot à trous dans l'alphabet {a,c,e,l,m,o,r}. Pour finir, on appelle *dictionnaire* une liste de mots (liste de listes de lettres).

1. Définir le prédicat `motTrouRef/1` tel que `motTrouRef(?M)` est satisfait si `M` représente le mot à trou "-m-r–".
2. Définir le prédicat `prefixe/2` tel que `prefixe(?L1, +L2)` est satisfait si `L1` est un préfixe de `L2`, c'est-à-dire une sous-liste non vide de `L2` commençant au premier élément.

   Ainsi on doit avoir

```
?- prefixe([a,m,i],[a,m,i,r,a,l]).          ---->   true.
?- prefixe([],[a,m,i,r,a,l]).               ---->   false.
?- prefixe([m,i],[a,m,i,r,a,l]).            ---->   false.
?- prefixe(L,[a,m,i]).                      ---->   [a] ;
                                                    [a,m] ;
                                                    [a,m,i].
```

3. Définir le prédicat `chercheDico/3` tel que `chercheDico(+Mot, +Dico, ?Res)` est satisfait si `Res` est la liste des mots du dictionnaire `Dico` dont le mot `Mot` est un préfixe.
   Ainsi on doit avoir

```
?- chercheDico([a,m,i],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],Res).
Res=[[a,m,i,r,a,l],[a,m,i,c,a,l]].
?- chercheDico([a,m,e,r],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],Res).
Res=[].
```

4. Définir le prédicat `valeurPossible/5` tel que `valeurPossible(?X,+Pref,+Dico,+Alphabet,?NewPref)` est satisfait si `X` est une lettre de `Alphabet` telle que le mot `NewPref`, obtenu en rajoutant `X` à la fin du mot `Pref`, est un préfixe d'au moins un des mots du dictionnaire `Dico`. Notez que si `X` était une variable libre, une valeur devrait lui être affectée pour pouvoir satisfaire ce prédicat.
   Ainsi on doit avoir

```
?- valeurPossible(r,[a,m,i],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                    [a,m,i,r,l,c,o],NewPref).
NewPref=[a,m,i,r].
?- valeurPossible(d,[a,m,i],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                    [a,m,i,r,l,c,o],NewPref).
false.
?- valeurPossible(X,[a,m,i],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                    [a,m,i,r,l,c,o],NewPref).
X=r,
NewPref=[a,m,i,r] ;
X=c,
NewPref=[a,m,i,c] ;
false.
?- valeurPossible(X,[a,m,i],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                    [a,m,i,l,o],NewPref).
false.
?- valeurPossible(X,[e],[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                    [a,m,i,r,l,c,o],NewPref).
false.
```

5. Définir le prédicat `remplit/3` tel que `remplit(+MT, +Dico, +Alphabet)` est satisfait s'il y a au moins un mot dans le dictionnaire `Dico` qui remplit le mot à trous `MT` dans l'alphabet `Alphabet`.
   Ainsi on doit avoir

```
?- remplit([t,X,t,Y], [[t,a,t,a],[t,o,t,o],[t,u,t,u],[t,o,t,a,l]], [t,a,o,l]).
X = Y, Y = a ;
X = Y, Y = o ;
false.
?- M = [l,_,s], remplit(M,[[l,a,s],[l,e,s,e],[l,i,t]],[l,s,t,e,i]).
false.
?- M = [l,_,s], remplit(M,[[l,a,s],[l,e,s,e],[l,i,t]],[l,s,t,e,i,a]).
M = [l,a,s] ;
false.
?- motTrouRef(MR), remplit(MR,[[a,m,i,r,a,l],[a,m,i,c,a,l],[a,m,o,r,a,l]],
                           [a,m,i,r,l,c,o]).
MR = [a,m,i,r,a,l] ;
MR = [a,m,o,r,a,l] ;
false.
```