16 [dupliquer] Programme qui crée (2e argument) une copie du terme passé en (1er) argument. Deux solutions : soit avec "=..", soit avec `functor` et `arg`.

17 [lit_assert] Faire une boucle qui lit des termes, et les « asserte ». Par convention, le terme stop ne sera pas asserté et correspondra à la fin de la boucle.

18 [member] Prédicat `member/2` qui teste l'appartenance d'un élément à une liste.

19 [delete] Prédicat `delete/3` qui supprime toutes les occurrences d'un élément d'une liste.

20 [append] Prédicat `append/3` qui concatène deux listes pour former une troisième (`append(`$L_1$`,`$L_2$`,`$L_3$`)` : $L_3 = L_1 + L_2$).

21 [tresser] Prédicat `tresser/3` qui crée une nouvelle liste W avec deux listes U et V de telle façon que W = $u_1$, $v_1$, $u_2$, $v_2$ etc.

22 [reverse] Prédicat `reverse/2` qui inverse les éléments d'une liste.

23 [avant] Il existe en Prolog un prédicat prédéfini `name/2` qui établit une correspondance entre un atome et la liste des lettres (codes ASCII) qui forme son nom.
Exemples :

```
| ?- name(machin,L).
L = [109,97,99,104,105,110] ?
yes
| ?- name(Mot,[65,66,67]).
Mot = 'ABC' ?
yes
```

Définir le prédicat `avant/2` tel que `avant(X,Y)` est vrai si X est avant Y dans l'ordre lexicographique.

24 [miroir] Prédicat `miroir/2` qui vérifie que deux mots sont des miroirs l'un de l'autre.

25 [sous_ens] Définir un prédicat `sous_ens`/2, tel que `sous_ens(X,Y)` est vrai si `X` et `Y` sont deux listes, et que tous les éléments de la liste `X` sont dans la liste `Y` (dans un ordre quelconque).

Que se passe-t-il si l'on appelle `sous_ens`/2 avec le premier argument non instancié ?

26 [sous_liste] Définir un prédicat `sous_liste`/2, tel que `sous_liste(X,Y)` est vrai si `X` et `Y` sont deux listes, et que la liste `X` se retrouve dans son intégralité, dans le même ordre et sans discontinuité, dans la liste `Y`.