Tableaux, exercice du Mastermind

Exemple

Proposition:

Vert	Bleu	Vert	Bleu	Noir
------	------	------	------	------

Solution:

Vert Noir Bleu Noir Rou	ge
-------------------------	----

Notre proposition a une couleur bien placée (le premier vert), et deux couleurs mal placées (le premier bleu et le noir). Le deuxième vert, ainsi que le rouge ne sont ni bien placés ni bien placés.

Proposition:

Vert Bleu Vert Bleu Noir

Solution:

Vert Noir	Bleu Noir	Rouge
-----------	------------------	-------

Pour trouver les bien placés:

Pour toute couleur de la proposition Si on a la même couleur à la même position dans la solution Effacer cette couleur dans la proposition et dans la solution

Il faut effacer les couleurs bien placées, pour ne pas les réutiliser plus tard.

Proposition:



Solution:



Pour trouver les mal placés:

Pour toute couleur de la proposition

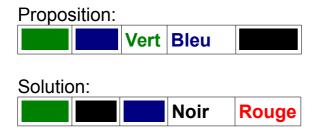
Chercher la première occurrence de cette couleur dans la solution Si on l'a trouvée, l'effacer et remonter au « pour »

lci à nouveau, il faut effacer les couleurs identifiées, pour ne pas les réutiliser plus tard.



On a trouvé le bleu dans les deux tableaux, donc on l'efface. Si on ne l'enlève pas, on le comptera une deuxième fois, car la proposition contient un deuxième bleu!

Ensuite, on a:



On a trouvé le noir dans les deux tableaux, donc on l'efface. On s'arrête là, à la première occurrence du noir, on ne regarde pas la deuxième occurrence du noir, pour ne pas le compter deux fois.

Note:

On ne peut pas réellement effacer un élément dans un tableau. Nous allons donc effacer « logiquement » les éléments. Pour ce faire, on emploie un tableau de booléens initialisé à *vrai*, qui dit si l'élément est valide (i.e. si il n'est pas supprimé).

Code:

```
module testerProposition(proposition, solution: tableau [1 à k] de Chaines,
                                        bienPlacés ↑, malPlacés ↑ : entiers)
     i : entier // index pour parcourir la proposition
     j : entier // index pour parcourir la solution
     propositionValide : tableau de booléens //pour les effacements
     solutionValide : tableau de booléens
                                               //pour les effacements
     propositionValide ← nouveau tableau [1 à k] de booléens
     solutionValide ← nouveau tableau [1 à k] de booléens
     initialiser(propositionValide, vrai) //tout mettre à vrai
     initialiser(solutionValide, vrai)
                                    //tout mettre à vrai
     bienPlacés ← 0
     malPlacés ← 0
     //Chercher les couleurs bien placées
     pour i de 1 à k faire
           si proposition[i] = solution[i] alors
                 bienPlacés ← bienPlacés + 1
                 propositionValide[i] ← faux //effacer
                 solutionValide[i] ← faux //effacer
           fin si
     fin pour
     //Chercher les couleurs mal placées
     pour i de 1 à k faire //on cherche la couleur proposition[i]
           trouvé ← faux //pour s'arrêter dès qu'on a trouvé la couleur
           i ← 1
           tant que NON trouvé ET j <= n faire
                 si proposition[i] = solution[j] ET propositionValide[i]
                                              ET solutionValide[j] alors
                       malPlacés ← malPlacés + 1
                       propositionValide[i] ← faux
                       solutionValide[i] ← faux
                       trouvé ← vrai
                 fin si
                 j ← j+1
           fin tant que
     fin pour
fin module
```

```
module initialiser(tab↑↓ : tableau [1 à k ] de booléens, val: booléen)

i: entier
    pour i de 1 à k faire
        tab[i] ← val
```

fin module

fin pour

Note: On peut améliorer un peu, en se passant du tableau propositionValide.