

Option Informatique en Spé MP et MP*

Programmation Caml : deux exercices

Placez au début de votre programme un commentaire indiquant votre nom, votre classe, la date, et le sujet du T.P. Rédigez les réponses aux questions autres que celles de pure programmation sur une feuille de papier séparée : vous n'êtes pas ici pour faire preuve de vos éventuels talents de dactylographe.

À la recherche des nombres d'Ulam

► L'ensemble des nombres d'ULAM est une partie de \mathbb{N}^* définie comme suit :

- 1 et 2 sont des nombres d'ULAM ;
- un naturel $u \geq 3$ est un nombre d'ULAM ssi u se décompose d'une et une seule façon en somme de deux nombres d'ULAM distincts.

Nous noterons u_n le n -ième nombre d'ULAM : $u_1 = 1, u_2 = 2$.

Question 1 • Vérifiez que $u_{10} = 18$ et déterminez u_{11} .

Question 2 • Montrez que l'ensemble des nombres d'ULAM est infini.

Question 3 • La suite de FIBONACCI est définie par $F_0 = 0, F_1 = 1$ et la relation de récurrence $F_{n+2} = F_{n+1} + F_n$ pour $n \in \mathbb{N}$. Établissez la majoration $u_n \leq F_{n+1}$.

Question 4 • Rédigez en Caml une fonction :

```
décompositions : int -> int list -> int
```

spécifiée comme suit : `décompositions n l` donne le nombre de façons de décomposer n en somme de deux éléments distincts de la liste l . Vous pourrez supposer que cette suite est sans doublon, et présentée en ordre (strictement) décroissant.

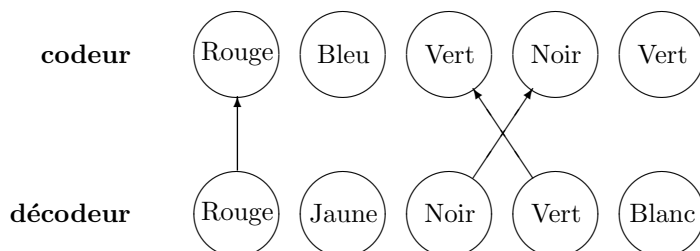
Question 5 • Rédigez en Caml une fonction :

```
nouvel_ulam : int list -> int list
```

spécifiée comme suit : notons s_n la liste présentée en ordre (strictement) décroissant des $n \geq 2$ premiers nombres d'ULAM ; appliquée à s_n , `nouvel_ulam` rend la liste s_{n+1} .

À propos du jeu *MasterMind*

► Dans le jeu *Master Mind*, le codeur choisit une combinaison de cinq fiches de couleur, qu'il tient cachée. Le décodeur doit deviner quelles sont ces fiches, en faisant des proposition successives : chaque proposition est elle aussi une combinaison de cinq fiches de couleur. Le codeur répond avec des témoins noirs et/ou blancs ; voyons ceci sur un exemple : le codeur a choisi la combinaison (Rouge,Bleu,Vert,Noir,Vert) et le décodeur propose (Rouge,Jaune,Noir,Vert,Blanc).



Le codeur répond (noir,blanc,blanc): le témoin noir correspond à la fiche Rouge; les deux témoins blancs correspondent aux fiches Noir et Vert de la proposition. De façon générale, le codeur pose un témoin noir pour chaque arc vertical; puis un témoin blanc pour chaque arc oblique. De chaque fiche de la proposition du décodeur part au plus un arc; et chaque fiche de la combinaison du codeur est visée par au plus un arc. Le jeu se termine lorsque la proposition du décodeur coïncide entièrement avec la combinaison du codeur, qui répond donc avec cinq témoins noirs!

► Notez bien que chaque *combinaison* (du codeur ou du décodeur) est ordonnée: si le décodeur propose ensuite (Rouge,Jaune,Vert,Noir,Blanc) il obtiendra une réponse différente, à savoir «trois témoins noirs». Il apprend ainsi que les couleurs aux positions 3 et 4 (en partant de la gauche) sont respectivement Vert et Noir.

Question 1 • Il y a huit couleurs: combien existe-t-il de combinaisons différentes?

► Nous définissons le type Caml suivant:

```
type couleur = Blanc | Jaune | Orange | Rouge | Marron | Vert | Bleu | Noir ;;
```

Question 2 • Rédigez en Caml une fonction:

```
compare : 'a list * 'a list -> int
```

spécifiée comme suit: `compare (u,v)` compare les deux listes u et v ; si elles sont de longueurs différentes, l'exception prédéfinie `Invalid_argument "compare"` est levée; sinon, notant n la longueur commune des deux listes, le résultat est le nombre d'indices $i \in \llbracket 0, n - 1 \rrbracket$ tels que $u_i = v_i$. Remarque: vous n'utiliserez ni fonction auxiliaire, ni `list_length`; vous filtrerez le couple (u, v) en examinant quatre motifs différents.

Question 3 • Réigez en Caml une fonction:

```
élimine : 'a -> 'a list -> 'a list
```

spécifiée comme suit: si la liste q contient au moins une occurrence de x , alors `élimine x q` élimine l'une de ces occurrences; sinon, l'exception prédéfinie `Not_found` est levée. Vous n'utiliserez pas de fonction auxiliaire; vous filtrerez la liste q en examinant trois motifs différents.

Question 4 • Rédigez en Caml une fonction:

```
réponse : 'a list -> 'a list -> int * int
```

spécifiée comme suit: `réponse u v` compare les deux listes u et v ; si elles sont de longueurs différentes, l'exception `Invalid_argument "compare"` est levée; sinon, le résultat est la réponse (n, b) du codeur lorsque u est sa combinaison, et v la proposition du décodeur. n est le nombre de témoins noirs, et b le nombre de témoins blancs.

Question 5 • Supposons que l'on joue avec deux couleurs et deux positions: déterminez le nombre *maximal* et le nombre *moyen* de propositions que devra faire le décodeur pour trouver la combinaison du codeur.

FIN