info401 : Programmation fonctionnelle
TP 3 : deux suites (presque) similaires

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Ce TP, comme les suivants, sera noté. Je ne demande aucun compte rendu à part, mais seulement un *unique* fichier Caml, commenté.

- Votre fichier doit contenir du code valide et ne doit pas provoquer d'erreur lorsqu'on l'évalue avec l'interprète Caml. (Testez avant de me l'envoyer : si votre fichier ne s'évalue pas correctement, vous perdez automatiquement 5 points sur votre note finale.)
- Votre fichier devra contenir un commentaire contenant votre nom, prénom et filière. (Idem, si ce n'est pas le cas, vous perdez 5 points sur votre note finale.)

Partie 1 : la suite de Conway

La suite de Conway est la suite suivante :
`1`
`11`
`21`
`1211`
`111221`
`312211`
`13112221`
...
Si vous ne connaissez pas cette suite, passez quelques minutes à essayer de deviner comment on la calcule avant de passer à la suite.

Pour calculer le terme suivant, on *lit* simplement le terme précédent :
- `1` donne "un `1`", soit `11`,
- `11` donne "deux `1`", soit `21`,
- `21` donne "un `2` et un `1`", soit `1211`,
- etc.

Si on part avec le terme `777777778888888`, on obtient :
`777777778888888`
`8778`
`182718`
`111812171118`
`3118111211173118`
...

Question 1. Programmez une fonction `conway : int list -> int -> int list` qui calcule le n-ème terme de la suite à partir d'une suite initiale d'entiers.

Attention : il est conseillé de découper votre code en plusieurs fonctions

Question 2. La taille du terme de la suite semble augmenter à chaque étape. Écrivez une fonction `conway_croissance : int list -> int -> float list` qui calcule la facteur de croissance de la suite.

Par exemple, `conway_croissance [1] 7` donnera `[2. ; 1. ; 2. ; 1.5 ; 1. ; 1.33333333]` car la taille des sept premiers termes est 1, 2, 2, 4, 6, 6, 8.

Que constatez-vous ?

Question 3. Écrivez une fonction de `conway_term` qui soit récursive terminale. Attention, cela signifie que toutes les fonctions (principale et auxiliaires) doivent être récursives terminales. En même temps, essayez de ne pas être trop inefficace en temps...

Vérifiez bien que vous obtenez la même chose qu'avec `conway`.

Partie 2 : la suite de Robinson

La suite de Robinson ressemble à la suite de Conway, à cela prêt qu'on compte le nombre total d'un chiffre dans la liste, et qu'on les compte par ordre croissant. Si on part de `1`, cela donne :

```
1
11
21
1112
3112
211213
312213
...
```

Par exemple, le passage de la ligne 3 à la ligne 4 se fait de la manière suivante : au total, `21` contient un "`1`" et un "`2`", soit `1112`. La ligne 7 est obtenue par : `211213` contient trois "`1`", deux "`2`" et un "`3`", soit `312213`.

Question 1. Programmez (de manière récursive terminale) la fonction `robinson : int list -> int -> int list`.

Question 2. Calculez `robinson [1] 1`, `robinson [1] 2`, `robinson [1] 3`, `robinson [1] 4`, `robinson [1] 5`, `robinson [1] 6`, ...

Que constatez-vous ?

Même question si on commence avec `[6 ;7]`.

Question 3. Comment pouvez-vous modifier votre code compter les nombre dans l'ordre décroissant ?

Que se passe-t'il ?

Question 4. La suite de Robinson est toujours *ultimement périodique*. Programmer une fonction qui donne :

- la période,
- le cycle,
- la pré-période (nombre d'itération necessaires avant de trouver le premier cycle),
- les termes précédents la première période.

Attention : essayez de ne pas faire trop de calculs. Vous pouvez utilisez les propriétés suivantes :

- lorsqu'on atteint la période, la taille du terme courant ne change plus,
- la période est forcément 1, 2 ou 3.

Partie 3 : Bonus

Reprogrammez le crible d'Eratosthene (fonction `crible2` ou `crible3` du TP 2) en faisant attention à n'utiliser que des fonctions recursives terminales.