Institut National Universitaire Champollion

Contrôle continu n°1

L3 Info
12 novembre 2018

PROGRAMMATION FONCTIONNELLE CAML
Durée : 1h30 - Aucun document autorisé

1. Quelques fonctions récursives simples

Dans toutes les questions n est supposé être un entier positif (sans qu'il soit besoin de le vérifier).

1. Écrire une fonction récursive $repet(n,p)$ construisant la liste constituée de n répétitions du nombre p.

```
#repet(3,6) ; ;
-  : int list = [6 ; 6 ; 6]
```

2. Écrire une fonction récursive $listeRepet(n)$ construisant la liste croissante constituée de i répétitions du nombre i pour i variant de 1 à n.

```
listeRep(6) ; ;
-  : int list
  =[1 ; 2 ; 2 ; 3 ; 3 ; 3 ; 4 ; 4 ; 4 ; 4 ; 5 ; 5 ; 5 ; 5 ; 5 ; 6 ; 6 ; 6 ; 6 ; 6 ; 6]
```

3. Écrire une fonction récursive $divEnl(n)$ construisant la liste décroissante de n à 0, où un terme est suivi de sa moitié s'il est pair et de son prédécesseur s'il est impair.

```
divEnl(81) ; ;
-  : int list = [81 ; 80 ; 40 ; 20 ; 10 ; 5 ; 4 ; 2 ; 1 ; 0]
```

2. Récursivité et chaînes de caractères : un cryptage simple

On pourra utiliser les fonctions tetec, tetes, reste et longChaine de la bibliothèque fournie : chaines.ml.
Le but de ce problème est d'écrire quelques fonctions de cryptage de chaînes de caractère, inspirées des premiers codes cryptés utilisés par les légions de Jules César.

1. Une première façon de crypter un mot est de mélanger ses lettres : Ecrire une fonction *melange(s)* qui permute deux par deux les lettres d'une chaîne de caractères.

```
 #melange("papa") ; ;
- : string = "apap"
#melange("maman") ; ;
- : string = "amamn"
```

2. Une autre façon classique de crypter un mot est de décaler ses lettres dans l'ordre de l'alphabet : Ecrire une fonction *décale(s,n)* qui retourne la chaîne obtenue en décalant chaque caractère de n positions dans la table ASCII.

```
#decale("bonjour",1) ; ;
-  : string = "cpokpvs"
#decale("bonjour",8) ; ;
- : string = "jwvrw}z"
```

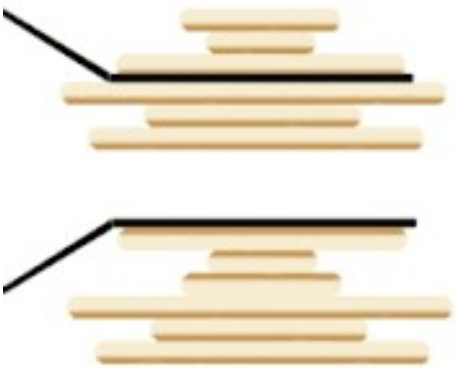
3. On obtient un cryptage assez efficace en combinant et en répétant les deux méthodes précédentes avec des paramètres différents (cf cryptographie Réseaux L3).
   Écrire une fonction $crypte(s,n)$ qui crypte la chaîne de caractère s en combinant :
   Un mélange puis un décalage de n positions puis un mélange puis un décalage de n-1 positions puis ... puis un mélange et un décalage de 1 position.

```
#crypt("Alesia, connais pas !",4) ; ;
- : string = "Kvo}sk6*myxxks}*zk}+"
```

Question subsidiaire : Quel était le prénom de Jules César ?

3. Récursivité et listes : le tri des galettes

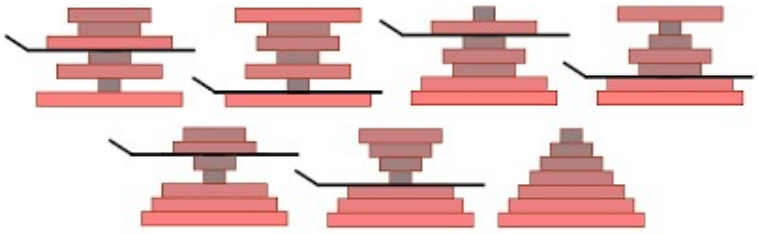
On dispose d'une pile de n galettes de diamètres différents, que nous souhaitons trier de manière à ce que la plus grande galette soit à la base de la pile et la plus petite au sommet.
Pour trier les galettes, nous disposons d'une spatule et nous ne pouvons utiliser qu'une seule opération : le retournement.
Pour effectuer un retournement, on insère la spatule entre deux galettes (ou sous la galette de la base), on place une main au sommet de la pile, et on retourne le paquet de galettes compris entre la spatule et le sommet.



On peut alors trier la pile de n galettes à l'aide de la méthode suivante :

```
 Repérer la plus grande galette.
 Si elle n'est pas déjà à la base de la pile
        Insérer la spatule sous cette galette et retourner
(la plus grande est alors au sommet)
        Retourner toute la pile
    (la plus grande est alors à la base)
  Poursuivre le tri avec les n-1 galettes supérieures
```

Voici un exemple de tri de galettes à l'aide de 6 retournements .



Nous représentons dans cet exercice, une pile de galette par une liste d'entiers 2 à 2 distincts, représentant les diamètres des galettes, du sommet de la pile jusqu'à sa base :

```
let l = [3 ;4 ;8 ;2 ;5 ;6 ;7 ;1] ; ;
```

1. Écrire une fonction *maxi*, qui calcule le maximum d'une liste d'entiers. (on définira localement la fonction *max* calculant le maximum de deux entiers.)

   ```
   # maxi [3 ;4 ;8 ;2 ;5 ;6 ;7 ;1] ; ;
   -  : int = 8
   ```

2. Ecrire une fonction *inverse*, qui, à une liste l associe la même liste mais dans l'ordre inverse.

   ```
   #inverse [3 ;4 ;8 ;2 ;5 ;6 ;7 ;1] ; ;
   -  : int list = [1 ; 7 ; 6 ; 5 ; 2 ; 8 ; 4 ; 3]
   ```

3. Ecrire une fonction *retourne*(l,p) qui retourne les éléments de la liste l compris entre le sommet de la liste et l'élément de valeur p (on considère que la tête de la liste est à l'indice 1).

   ```
   retourne ([1 ;2 ;3 ;4 ;5 ;6],3)  ; ;
   -  : int list = [3 ; 2 ; 1 ; 4 ; 5 ; 6]


   retourne  (l,maxi l)  ; ;
   -  : int list = [8 ; 4 ; 3 ; 2 ; 5 ; 6 ; 7 ; 1]
   ```

   (On pourra utiliser une fonction récursive auxiliaire).

4. Écrire enfin une fonction *triGalette*, triant la pile de galette à l'aide de l'algorithme vu plus haut.

   ```
   triGalette l ; ;
   -  : int list = [1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8]
   ```