ANNÉE UNIVERSITAIRE 2012/2013
1ÈRE SESSION

Parcours : CSB5
Code UE : INF353
Épreuve : Programmation 3
Date : 28 janvier 2013
Heure :
Durée : 1h30
Épreuve de Mme Irène Durand

Le barème est donné à titre **indicatif**. Le sujet comporte 2 pages.

Exercice 1 *(2pts)*

Évaluer les expressions suivantes :

1. `(cons '() '(1))`
2. `(list '() '(1))`
3. `(append '() '(1))`
4. `(cons '() 1)`

Exercice 2 *(6pts)*

Soit la fonction mathématique *factorielle (n)* (notée $n!$) définie pour un entier positif n par

$$\begin{cases} 0! = 1 \\ n! = n(n-1) \quad \textit{pour } n > 0 \end{cases}$$

1. Écrire une version naïve de la fonction `factorielle (n)` calquée sur la définition.
2. Cette version est-elle *récursive terminale* ?
3. Écrire une version récursive terminale de la fonction factorielle.

Exercice 3 *(4pts)*

Écrire une fonction `expand (v)` qui prend en paramètre un vecteur d'entiers et qui retourne une liste contenant, pour chaque case d'indice i du tableau, une liste contenant $v[i]$ fois l'entier i. Exemples :

```
CL-USER> (expand #())
NIL
CL-USER> (expand #(2 4 1))
((0 0) (1 1 1 1) (2))
CL-USER> (expand #(2 0 4 1))
((0 0) NIL (2 2 2 2) (3))
```

```
(defparameter *sapin*
  #(#(0 0 0 0 0 0 0 1 1)
    #(0 0 0 0 1 0 0 0 1)
    #(0 0 0 1 1 1 0 0 0)
    #(0 0 1 1 1 1 1 0 0)
    #(0 1 1 1 1 1 1 1 0)
    #(0 0 0 1 1 1 0 0 0)
    #(0 0 0 1 1 1 0 0 0)
    #(0 0 0 0 0 0 0 0 0)))
```

```
CL-USER> (print-image *sapin*)
       @@
      @   @
     @@@
    @@@@@
   @@@@@@@
     @@@
     @@@

NIL
```

FIG. 1 – Représentations d'un sapin

Exercice 4 *(8pts)*

On considère des images noir et blanc de dimensions `h x w`. Une telle image sera représentée par un tableau contenant `h` lignes, chaque ligne étant elle-même représentée par un tableau de `w` entiers (`1` pour noir et `0` pour blanc). Par exemple, la variable `*sapin*` définie Figure 1 contient la représentation d'une image `8 x 9`.

1. Écrire une fonction `print-image (image)` qui imprime sur la sortie standard l'image en imprimant un espace pour les bits à 0 et un `@` pour les bits à 1. Par exemple, l'image représentée dans la variable `*sapin*` doit s'imprimer comme montré Figure 1.
2. Que donne l'appel suivant ?

   ```
   (print-image #(#(1 1 0) #(0 1 0) #(0 1 1)))
   ```

3. Modifier la fonction `print-image` de manière à pouvoir passer dans un paramètre optionnel le caractère à afficher.

En vue de compresser plus facilement une image `h x w`, on souhaite transformer sa représentation en un vecteur de taille `h x v` dans lequel les lignes sont stockées les unes à la suite des autres.

4. Écrire une fonction `image2vector (image)` qui prend en paramètre une image représentée par un tableau de lignes et qui retourne l'image représentée dans un vecteur. Exemple :

   ```
   CL-USER> (image2vector *sapin*)
   #(0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0
     0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0)
   ```

FIN