# INSAT
# MPI
# Structures de données et algorithmique
# Série 1

**Exercice 1**

Ecrire une fonction récursive permettant de convertir un nombre N donné (en base 10) en une base b ($2 \leq b \leq 10$)

**Exercice 2**

Sur les chaînes de caractères, on dispose des 3 opérations suivantes :

- *Dernier* : **Chaine** $\rightarrow$ **Caractere**
- *Debut* : **Chaine** $\rightarrow$ **Chaine**
- *AjoutF* : **Chaine** x **Caractere** $\rightarrow$ **Chaine**

• L'opération *Dernier* délivre le dernier caractère de la chaîne passée en argument.
• L'opération *Début* délivre la chaîne passée en argument privée de son dernier élément.
• L'opération *AjoutF* délivre la chaîne passée en argument à laquelle on a rajouté à la fin le caractère donné en argument.

Dans la suite la chaîne vide sera notée « ChVide »

1 – Construire une fonction récursive qui calcule la longueur d'une chaîne.
2 – Construire une fonction récursive qui teste si une chaîne1 est extraite d'une chaîne2, i.e. les caractères de chaîne1 sont présents (dans l'ordre mais pas nécessairement de façon contiguë) dans chaine2 .
Exemples : *EstExtraite*('ABC', 'KVABTC')=Vrai
*EstExtraite*('ABC','ANMCB')=Faux

Dans la suite, on supposera que les chaînes décrivent des représentations décimales (en base 10) d'entiers.

3 - Ecrire une fonction récursive qui teste si une telle chaîne est croissante (i.e. les chiffres se présentent dans la chaîne dans un ordre croissant)

Exemples : *Croissante*('2468')=Vrai
*Croissante*('2466')=Vrai
*Croissante*('2168')=Faux

4 – Ecrire une fonction récursive qui construise le successeur d'un entier donné.
Exemples : *Succ*('2468')='2469'
*Succ*('78299')='78300'

N.B. Pour les questions 3 et 4 on pourra utiliser les deux fonctions inverses suivantes :

*Int* : Caractere → Chiffre

*Caract* : Chiffre → Caractere

**Exercice 3**

On se propose d'étudier le jeu décrit par les règles suivantes :

On dispose de n jetons réversibles, alignés. Chaque jeton à une face marquée 1 et une face marquée 0. Au départ seules les faces 0 des n jetons sont visibles. Le but du jeu est de retourner les différents jetons de façon que les seules faces visibles soient les 1 (cf. figure 1)

| | |
|---|---|
| No. Jeton | 123………….n |
| Configuration initiale | 000………….0 |
| Configuration finale | 111………….1 |

Figure1 : Etats du baguenaudier

Le retournement des jetons obéit aux règles suivantes :

R1 : On peut toujours retourner le premier jeton (celui qui est le plus à gauche)

R2 : On peut retourner le Ième jeton, pourvu que soient visibles les faces 0 des (i-2) premiers jetons et la face 1 du (I-1) ème jeton

Ecrire un algorithme récursif permettant de passer de la configuration initiale à la configuration finale en respectant les règles précédentes. Pour écrire cet algorithme on pourra introduire deux fonctions mutuellement récursives :

- Une fonction notée *Bag(jet,k)* qui transforme un tableau de n jetons dont les k premiers sont à 0 en un nouveau tableau dont les k premiers sont à 1.
- Une fonction notée *Debag(jet,k)* qui transforme un tableau de n jetons dont les k premiers sont à 1 en un nouveau tableau de n jetons dont les k premiers sont à 0.

On définit une fonction *Baguenaud* par :

*Baguenaud(Jet,k,b)* = **si** b **alors** *Bag(jet,k)*
**Sinon** *Debag(jet,k)*
**Fsi**

Exprimer l'algorithme récursif de la question précédente à l'aide de *Baguenaud.*