

Algorithmique et structures de données II**TD1 : Récursivité****Exercice 1**

Ecrire une fonction récursive en langage C qui, étant donné un entier positif n , renvoie la somme des n premiers entiers.

Exercice 2

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

Exercice 3

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

- *Dernier* : **Chaîne** \rightarrow **Caractere**

- *Debut* : **Chaîne** \rightarrow **Chaîne**

- *AjoutF* : **Chaîne** X **Caractere** \rightarrow **Chaîne**

- 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 – Ecrire en C une fonction récursive qui calcule la longueur d'une chaîne.

2 – Ecrire en C 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 chaîne2.

Exemples :

EstExtraite('ABC', 'KVABTC')=Vrai

EstExtraite('ABC', 'ANMCB')=Faux

Dans la suite, on supposera que les chaînes décrivent des représentations décimales (en base10) d'entiers.

3 – Ecrire en C 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 : Caractère → Chiffre

char : Chiffre → Caractère

Exercice 4

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

No. Jeton 123.....*n*

Configuration initiale 000.....0

Configuration finale 111.....1

Le retournement des jetons obéit aux règles suivantes :

Règle 1 : On peut toujours retourner le premier jeton (celui qui est le plus à gauche)

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

Pour passer de la configuration initiale à la configuration finale en respectant les règles précédentes, 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.

Ecrire en langage C les fonctions ***Bag(jet, k)*** et ***Debag(jet, k)***.