Licence Informatique — **UE INF101**
Licence Mathématiques — **Initiation à la programmation**
Université Bordeaux 1 — **Travaux Dirigés**
2006-2007 — **feuille 8**

Exercice 1. Factorielles

1. Écrire une fonction itérative `unsigned long long fact_iter (int n)` qui calcule et retourne `n!`.
2. Écrire une fonction récursive `unsigned long long fact_rec (int n)` qui calcule et retourne `n!`.
3. Écrire un programme de test de ces deux fonctions.
   Question à traiter en tp : A partir de quelle valeur de n, le calcul est-il incorrect ? Pourquoi ?

Exercice 2. L'algorithme d'Euclide permet de calculer le PGCD de deux entiers :

Le PGCD de a et b est égal à b si ce dernier divise a, sinon le PGCD de a et b est égal au PGCD de b et de r (où r représente le reste de la division euclidienne de a par b).

Écrire une fonction récursive `unsigned pgcd(unsigned a, unsigned b)` utilisant cet algorithme.

Exercice 3. Écrire une fonction récursive qui retourne la valeur du plus grand élément d'un tableau contenant n nombres entiers distincts.

Exercice 4. Écrire une fonction récursive qui insère une chaîne donnée s à la position k (indice) d'un tableau contenant n chaines de caractères ($0 \leq k \leq n$). La taille du tableau sera strictement supérieure à n.

Exercice 5. Écrire une fonction récursive qui teste si un tableau de chaînes est trié dans l'ordre lexicographique ou non. Cette fonction retournera un booléen (true ou false suivant le cas).

Exercice 6. On considère le code source `factorielle2.c`. Ce programme fournit une trace des appels récursifs. Étudier le source. Compiler et exécuter ce programme.

Exercice 7. Le nombre de combinaisons de `n` éléments pris `p` à `p` est défini récursivement par :

$C_n^0 = C_n^n = 1$
$\forall p, 0 < p < n \;\; C_n^p \;=\; C_{n-1}^{p-1} \;+\; C_{n-1}^p$

1. Écrire une fonction récursive `unsigned combinaison(int n, int p)` qui calcule et retourne C_n^p.
2. Modifier la fonction précédente afin de compter le nombre d'appels récursifs nécessaires au calcul de C_n^p. (On vérifiera que le nombre d'appels nécessaires au calcul de C_n^p est $2 * C_n^p - 1$)
3. Modifier la fonction précédente afin d'afficher une trace des appels récursifs nécessaires au calcul de C_n^p.
4. Ajouter à la fonction précédente une variable permettant de connaître la profondeur maximale des appels récursifs.((On vérifiera que la profondeur maximale des appels nécessaires au calcul de C_n^p est $n - 1$)

La commande combinaison fournie répond aux différentes questions de cet exercice.

Exercice Complémentaire 1. Une fonction récursive terminale est une fonction récursive telle qu'aucune opération n'est exécutée après l'appel récursif (la fonction `fact_rec` de l'exercice 1 n'est pas récursive terminale car le produit `n * fact_rec(n-1)` ne peut être effectué qu'après le retour de `fact(n-1)`). Écrire une fonction récursive terminale `unsigned long long fact_rec_term(int n,unsigned long long acc)` qui calcule et retourne `n!`.

Exercice Complémentaire 2. Écrire une fonction de recherche dichotomique dans un tableau d'entiers trié. Cette fonction retournera l'indice de l'élément cherché si celui ci existe dans le tableau ou -1 sinon. On fournira une version itérative et une version récursive.