Programmation : TP 5

Objectifs : [C] Fonctions récursives, récursivité terminale.

Quelques consignes

Ecrivez les différentes fonctions demandées en C dans un fichier `tp5.c`. Rappel : en fin de TP, envoyez vos fichiers par mail à `barnier@recherche.enac.fr`.

Ne pas programmer dans le style impératif dans ce TP. En conséquence, ne pas utiliser de boucle `for` ou `while` (sauf lorsque c'est explicitement demandé pour l'obtention des temps de calcul). Rappel : utiliser l'option `-W -Wall` pour obtenir tous les *warnings* en compilant avec *gcc*, et l'option `-O2` pour optimiser les appels récursifs terminaux.

[C] PGCD de deux nombres

Écrire une fonction qui calcule le PGCD (Plus Grand Diviseur Commun) de deux nombres puis une fonction `main` qui appelle cette fonction (en utilisant le tableau `argv` pour les arguments)
Indication : Utiliser l'opérateur infixe `%` pour obtenir le reste de la division euclidienne.

[C] Calcul de b^n

1. Construire une fonction `exponentielle` qui calcule b^n :
   (a) par récursion directe ;
   (b) par itération (récursivité terminale) ;
   (c) en utilisant le fait que $b^n = (b^2)^{n/2}$ quand n est pair (non nécessairement récursive terminale).
2. Dans les trois cas, évaluer le nombre d'opérations arithmétiques et la mémoire nécessaires.
3. Comparez les temps de calcul entre ces trois cas en compilant avec l'option d'optimisation `-O`. Utilisez la fonction `clock` de la librairie `time` (ne pas oublier l'`include` en début de fichier), qui renvoie le temps d'utilisation du processeur depuis le premier appel à cette fonction.

   ```
   printf("temps de calcul : %f\n", (float)(clock())/CLOCKS_PER_SEC);
   ```

   Indications : Si vous ne faites qu'un seul appel à la fonction exponentielle, les écarts de temps de calcul ne seront pas perceptibles. Encadrez cet appel par une boucle comprenant un grand nombre d'itérations. Prendre également des valeurs de n suffisamment grande pour observer le gain obtenu par la méthode (c).

[C] Nombre premier

Construire une fonction `premier` qui teste si un nombre est premier.

[O'Caml, C] Comparaison des temps de calcul en C et O'Caml

1. Écrire en O'Caml (et tester sous le toplevel) une fonction calculant b^n comme au **??**.
2. Utiliser cette fonction dans un programme (fichier source `tp5.ml`) affichant le résultat et le temps de calcul.

   Indication : Le temps d'utilisation du processeur peut être obtenu avec la fonction `time` (qui retourne un `float`) de la librairie `Sys` :

   ```
   let t = Sys.time () ...
   ```

3. Compilez-le en mode optimisé, avec la commande :

   ```
   ocamlopt -o tp5.opt tp5.ml
   ```

4. Comparez avec C.