Mercredi 31 Mai 2017 - 9h30-12h30 - AMPHI ASTIER

Responsable : Annick Valibouze

- Examen de Programmation - ISUP1 - Université Paris 6

Aucun document n'est autorisé. Cet examen est composé de 5 parties sur 4 pages à difficultés croissantes. Les questions plus difficiles donnent lieu à plus de points que les faciles.

1. Questions "faciles"

a) Qu'affiche le programme suivant ? (aucune explication n'est demandée)

```
void main(){
   int x=0;
   printf(''f1(x)=%d'',f1(x));
   printf(''x=%d \n'',x);
   x=0 ;
   printf(''f2(&x)=%d'',f2(&x));
   printf(''x=%d \n'',x);
   while(x< 5) {f3(x) ; f4(x);}}

int f1(int x) {x=x+2 ; return x ;}
int f2(int * x) {*x=*x+2 ; return *x ;}
int f3(int x) {int x=0 ; x=x+1 ; if (x < 4) printf(''Dans f3 x=%d '',x);}
int f4(int x) {static int x=0; x=x+1;
               if (x < 4) printf(''Dans f4 x=%d'',x);}
```

a) Le but de cet exercice est de calculer efficacement une suite de factorielles pour des entiers croissants. Ecrire une fonction `int fac(int n)` qui calcule la factorielle d'un entier `n` et qui, si la factorielle d'un entier `m` inférieur à `n` a déjà été calculée, s'en sert. Exemple : si `fac(4)` est exécutée alors le résultat 24 est sauvegardé jusqu'au prochain appel qui, s'il est `fact(6)`, ne calculera que $24 \cdot 5 \cdot 6$. (Aucune variable globale ne doit être utilisée.)

b) Ecrire une fonction `ProdChi` qui calcule le produit des chiffres d'un nombre écrit en base 10. Exemple : `ProdChi(251)` retournera 10.

c) Ecrire une fonction `Decomp` qui prend en arguments deux entiers positifs m et $b < 10$ et qui retourne la représentation de m en base b. Vous avez libre choix pour la représentation des données. Exemple : `Decomp(11,2)` retournera les coefficient des puissances de 2 dans $11 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^3$. Indication : utiliser récursivement la division euclidienne par b.

d) En supposant écrite la **fonction** du c), écrire un **programme** `Decomp` dont les paramètres doivent être deux entiers positifs m et $b < 10$ (pour l'utilisateur) et qui affiche la représentation de m en base b. Exemple : la commande `Decomp 11 2` affichera 1 1 0 1.

e) Ecrire une fonction récursive `fib` en `C` qui calcule $fib(n)$ pour tout entier positif n où fib est la fonction de fibonacci d'un entier :

$$\begin{aligned} fib(0) &= fib(1) = 1 \quad , \\ fib(n+1) &= fib(n) + fib(n-1) \qquad si \quad n > 0 \, . \end{aligned}$$

f) Dans votre fonction `fib`, calculez-vous plusieurs fois la même chose ? (expliquez en une phrase). Que proposez-vous pour y remédier ? Ecrivez la nouvelle fonction.

2. Problème sur les tableaux à deux dimensions

Soit p un entier premier. Nous notons Z/pZ le corps des entiers modulo l'entier p représenté par $\{0, 1, \ldots, p-1\}$. Le but est d'écrire une fonction `tablemult` qui retournera la table de multiplication dans Z/pZ. Par exemple, pour $p = 3$, nous cherchons à calculer un tableau à deux dimensions représentant la table suivante :

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

a) Pour construire de telles tables, vous avez le choix entre une structure de tableau dans la pile, dans la zone statique ou bien dans le tas. Que choisissez-vous et pourquoi ?

b) Écrire la fonction `tablemult` selon votre choix précédent.

c) Faire un appel à votre fonction pour calculer la table de multiplication dans $Z/3Z$.

3. Chaînes de caractères

Si vous préférez les listes aux chaînes de caractères, ne répondez à aucune question de cette section et allez directement à la section suivante. Attention : certaines questions sont plus difficiles à traîter sur les listes que sur les chaînes de caractères.

Nous considérons les mots composés des trois lettres a, b et c, tels *aababc*, que nous représentons par des chaînes de caractères.

a) Ecrire une fonction `Occurence` qui prend en arguments une lettre et un mot et qui retourne le nombre d'occurences de cette lettre dans ce mot.

b) Ecrire une fonction `LongueurMot` qui calcule la longueur d'un mot (sans utiliser la fonction `C` qui calcule cette longueur).

c) Ecrire une fonction `LongueurEnTete` qui prend en arguments une lettre et un mot et retourne 0 si le mot ne commence pas par cette lettre et sinon retourne la longueur de la première séquence de cette lettre dans le mot.

d) Ecrire une fonction `LongueurEnQueue` qui prend en arguments une lettre et un mot et retourne 0 si le mot ne se termine pas par cette lettre et sinon retourne la longueur de la dernière séquence de cette lettre dans le mot. Cette fonction ne devra pas utiliser la précédente.

e) Ecrire une fonction **récursive** appelée `EgalMot` qui (sans utiliser la fonction C qui le fait) teste si deux mots sont identiques.

f) Ecrire en **itératif** la même fonction que e).

h) Ecrire une fonction `Renverse` qui retourne l'inverse d'un mot passé en paramètre.

i) Un palindrome est un mot qui se lit dans les deux sens comme aba, abba, cccbbaabbccc. Ecrire une fonction `Palindrome1` qui prend un mot en argument et teste s'il est un palindrome MAIS sans le renverser. On pourra utiliser les fonctions précédentes même si elles ne sont pas écrites.

j) Ecrire une fonction `Palindrome2` qui teste si un mot est palindrome en le renversant.

4. Listes à la place des chaînes de caractères

Pour ceux et uniquement ceux qui n'ont répondu à aucune question de la section 3 précedente. Les points entre ces deux sections ne sont pas cumulables. Traiter toutes les questions de la section 3 avec des listes de caractères représentant les mots. Notez que vous devrez définir un nouveau type pour les listes de caractères.

5. ECHAPPEMENTS (`setjmp` ET `longjmp` ET POINTEURS DE FONCTIONS)

Supposons que `liste` soit le type liste d'entiers définit en cours. Nous utilisons la fonction `cons` et les PSEUDO-FONCTIONS d'extractions `VAL` et `RESTE` du cours. Soit la fonction `map` suivante :

```
liste map(liste l, int (*f)(int)){
   if (l==NULL) return l;
   else return cons(f(VAL(l)),map(RESTE(l),f));}
```

a) Que fait la fonction map ?

b) Modifier la fonction `map` en

```
liste map2(liste l, int (*f)(int), int (*test)(int))
```

telle que si `test(f(VAL(l)))` retourne la valeur "faux" du `C` alors un échappement est provoqué (lire la question c) qui est liée)

c) Ecrire une fonction

```
liste map1(liste l, int (*f)(int), int (*test)(int))
```

qui appelle la fonction `map2`. La fonction `map1` retourne la même chose que `map` si aucun échappement n'est provoqué et, dans le cas contraire, provoque la sortie du programme avec un message d'erreur "calcul impossible".