

Examen facultatif, Programmation I

Sciences et Technologies du Vivant, Semestre 1

Chimie et Génie Chimique, Semestre 3

Mercredi 13 décembre 2006

1. N'oubliez pas de mettre vos NOM, PRÉNOM et SECTION sur toutes vos copies.
2. Merci de garder la donnée.
3. Vous pouvez rédiger vos réponses au crayon.
4. Vous n'avez pas à écrire un programme entier avec `#include`, etc. Rédigez uniquement la partie demandée. Dans le cas d'une fonction, écrivez l'en-tête et le corps de la fonction uniquement.
5. Les réponses qui font ce qui est demandé mais qui sont trop compliquées n'obtiendront pas le maximum des points.
6. Les nombres de points pour chaque question sont donnés dans la marge de droite, et sont à titre indicatif. Le total de points est sur 100.
7. La question 4c est une question bonus (au-delà des 100 points).

Question 1 – Commandes Unix 15 points

- (a) Un ami vous a parlé de la commande Unix `find` qui permet de faire des recherches sur le système de fichier. Vous aimeriez l'utiliser mais ne savez pas comment elle fonctionne. Ecrivez la commande Unix qui pourrait vous renseigner. (3)
- (b) La commande `ls -l` produit le listing suivant d'un répertoire : (3)
- ```
-rw-r--r-- 1 jberclaz cvlab 582 Dec 5 17:15 fichier.txt
-rw-r--r-- 1 jberclaz cvlab 155295 Dec 5 17:15 image.jpg
dr-xr-xr-x 2 jberclaz cvlab 4096 Dec 5 17:17 rep
drwxr-xr-x 2 jberclaz cvlab 4096 Dec 5 17:17 rep2
-rw-r--r-- 1 jberclaz cvlab 5418984 Dec 5 17:15 video.avi
```
- Ecrivez les commandes Unix nécessaires pour copier le fichier `fichier.txt` dans le répertoire `rep`.
- (c) Toujours dans le même répertoire qu'au point précédent, créez un nouveau sous-répertoire nommé `perso` dans le répertoire `rep2` en utilisant une seule commande. Assurez-vous que seulement vous et les membres de votre groupe puissiez y accéder et lire ce qui s'y trouve. (3)
- (d) Déplacez le fichier `fichier.txt` se trouvant dans `rep`, dans votre répertoire `perso`. (3)
- (e) Vous n'aimez pas le nom de votre répertoire `perso`. Renommez-le en `personnel`. (3)

## Question 2 – Fonctions et Booléens ..... 20 points

- (a) On suppose qu'on dispose de la fonction (2)
- ```
int pgcd(int a, int b)
```
- qui renvoie le PGCD de deux nombres `a` et `b`. Écrivez la fonction d'en-tête
- ```
bool premiers_entre_eux(int a, int b)
```

qui renvoie `true` si `a` et `b` sont premiers entre eux, `false` sinon. On rappelle que deux nombres sont premiers entre eux si et seulement si leur PGCD vaut 1. Écrivez la fonction `premiers_entre_eux` en une ligne.

- (b) Toujours en une ligne, écrivez la fonction d'en-tête (3)

```
bool premiers_entre_eux(int a, int b, int c)
```

qui renvoie `true` si `a`, `b` et `c` sont premiers entre eux, `false` sinon. Trois nombres sont premiers entre eux si et seulement si leur PGCD vaut 1. Comme on ne dispose que de la fonction `pgcd` qui permet de calculer le PGCD de deux nombres seulement, on utilisera la propriété suivante :  $pgcd(a, b, c) = pgcd(pgcd(a, b), c) = pgcd(a, pgcd(b, c))$ .

- (c) La fonction précédente n'est pas optimale : On sait que si `a` et `b` sont premiers entre eux, alors `a`, `b` et `c` sont premiers entre eux. On n'a alors besoin d'appeler la fonction `pgcd` qu'une seule fois. Comment écrire la fonction (4)

```
bool premiers_entre_eux(int a, int b, int c)
```

pour qu'elle n'appelle qu'une fois `pgcd` si `a` et `b` sont premiers entre eux, et deux fois sinon ? Pour cette fonction, vous n'êtes pas limités à une seule ligne.

- (d) On veut maintenant écrire la fonction (5)

```
bool multiple(int a, int b)
```

qui renvoie `true` si `a` est un multiple de `b`, et `false` sinon. On rappelle que `a` est un multiple de `b` si il existe un entier naturel `t` tel que  $a = bt$ . Comment écrire la fonction `multiple` sans utiliser l'opérateur modulo (`%`), mais avec une boucle `while` ?

- (e) Cette fois, on ne veut utiliser ni modulo ni boucle `while`. On utilisera pour cela la propriété suivante :  $multiple(a, b) = divide(a, b, a)$ , où la fonction `divide` est définie ainsi : (6)

$$divide(x, y, z) = \begin{cases} \text{faux} & \text{si } z = 0 \\ \text{vrai} & \text{si } z \neq 0 \text{ et } x = yz \\ divide(x, y, z - 1) & \text{si } z \neq 0 \text{ et } x \neq yz \end{cases}$$

Écrivez maintenant la fonction `multiple` et la fonction `divide`, qui sera une fonction récursive.

### Question 3 – Tableaux .....35 points

Les tableaux en C ont un inconvénient : ils ne connaissent pas leur propre taille et le programmeur doit toujours déclarer une variable ou constante séparée pour mémoriser la taille du tableau.

Vous allez écrire plusieurs fonctions de manipulation de tableaux d'entiers permettant de se libérer de cette contrainte en stockant la taille dans le premier élément du tableau lui-même. Ainsi, pour un tableau de  $n$  éléments, on allouera un tableau de  $n + 1$  éléments, dont l'élément 0 contiendra la valeur  $n$ , et les éléments de 1 à  $n$  contiendront les éléments du tableau.

**Exemple :** Le tableau standard `t` de 5 éléments

|      |   |   |   |   |   |
|------|---|---|---|---|---|
| i    | 0 | 1 | 2 | 3 | 4 |
| t[i] | 8 | 7 | 2 | 2 | 9 |

sera représenté par le tableau `s` de 6 éléments suivant :

|      |   |   |   |   |   |   |
|------|---|---|---|---|---|---|
| i    | 0 | 1 | 2 | 3 | 4 | 5 |
| s[i] | 5 | 8 | 7 | 2 | 2 | 9 |

Implémentez les fonctions suivantes qui permettront de gérer des tableaux d'`int`. Vous pouvez utiliser dans une fonction une fonction que vous avez implémentée précédemment : par exemple, vous pouvez appeler la fonction `lire` à l'intérieur de la fonction `somme`. L'en-tête de la première fonction vous est donné, vous devrez trouver les autres.

- (a) La fonction d'en-tête (4)

```
int * alloue_tableau(int taille)
```

prend comme argument la *taille* du tableau et retourne un pointeur sur le nouveau tableau. La fonction doit prendre soin d'allouer correctement la mémoire pour le nouveau tableau, ainsi que de stocker sa taille au bon endroit ;

- (b) `libere_tableau` prend en argument un pointeur sur un tableau d'entier et se charge de le désallouer ; (3)

- (c) `taille`. Cette fonction reçoit en paramètre un pointeur sur le tableau et retourne la taille du tableau ; (3)

- (d) `lire`. Cette fonction prend comme argument un pointeur sur le tableau ainsi que la position à lire. Elle retourne la valeur de l'élément désiré ou 0 si l'on tente de lire en dehors des limites du tableau. La position dans le tableau est exprimée de manière standard, c'est-à-dire de 0 à  $n - 1$  pour un tableau de taille  $n$ . (5)

**Exemple :** pour lire la première valeur du tableau représenté par `s` ci-dessus et la stocker dans une variable entière `n`, on écrira :

```
int n = lire(s, 0);
```

`n` doit alors contenir 8.

- (e) `somme`. Cette fonction doit retourner la somme des éléments d'un tableau passé en paramètre. Bien sûr, la taille du tableau stockée dans l'élément 0 ne doit pas être comprise dans la somme. (8)

- (f) `fusion`. La dernière fonction se charge de *fusionner* deux tableaux triés par ordre croissant dans un nouveau tableau. Elle prend comme paramètres deux pointeurs sur tableaux, et retourne un pointeur sur le nouveau tableau fusionné. Elle doit tout d'abord allouer de la mémoire pour un nouveau tableau, dont la taille sera la somme des tailles des deux tableaux à fusionner. Ensuite, la fonction doit copier les valeurs des deux tableaux à fusionner dans le nouveau tableau, en prenant soin que celui-ci soit aussi trié par ordre croissant. (12)

**Exemple :** soit les deux tableaux triés `t1` et `t2` :

```
t1 | 1 | 3 | 5 | 7 | 9 |
t2 | 2 | 4 | 6 |
```

Leur fusion donnera le tableau `t3` suivant :

```
t3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 |
```

On pourra utiliser la fonction `ecrire` qui permet d'écrire une valeur dans un tableau sans avoir à l'implémenter. `ecrire` prend trois paramètres : un pointeur sur le tableau, la position du tableau à modifier et la nouvelle valeur à écrire. Par exemple :

```
ecrire(s, 1, 0);
```

met 0 à la place du 7 dans le tableau `s`.

## Question 4 – Pointeurs ..... 50 points

- (a) Soit le morceau de code suivant : (10)

```
int *t;
int *n;
t = new int[5];
n = t + 2;
```

```

for (int i=0; i<5; i++)
 t[i] = 2 * i;
*n = 20;
t[3] = int(&n);

```

Complétez le tableau ci-dessous afin qu'il représente l'état de la mémoire à la fin du programme.

Chaque case du tableau représente 4 *octets*, c'est-à-dire l'emplacement mémoire nécessaire pour stocker un `int` ou une adresse mémoire. La première rangée contient les adresses mémoire. La deuxième rangée – que vous devrez compléter – représente le contenu de la mémoire alors que la dernière rangée affiche l'emplacement des variables fixées lors de la compilation.

On suppose que le système d'exploitation place le début du tableau `t` alloué dynamiquement à l'adresse 28.

|   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |     |
|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| 0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 | 52 | 56 | ... |
|   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |     |
| t | n |   |    |    |    |    |    |    |    |    |    |    |    |    |     |

- (b) Lisez attentivement le code ci-dessous et indiquez ce qu'il affiche dans le terminal : (20)

```

int i = 11;
int j = 17;

int *p = &i;
int *q;

*p = j * 2;

cout << "1. " << *p << endl;

cout << "2. i: " << i << ", j: " << j << endl;

i = int(&j);
q = (int *)(*p);
*q -= 5;
*p = *q * 2;

cout << "3. i: " << i << ", j: " << j << endl;

q = (int *)(*p);
j = 5;
p = p - 2;

cout << "4. i: " << i << ", j: " << j << ", q: " << int(q) << endl;

```

- (c) Lisez attentivement le code ci-dessous et indiquez ce qu'il affiche dans le terminal : (20)

```

int *t = new int[3];

int **p = new int*[3];

for (int i=0; i<3; i++) {
 p[i] = t + i;
 t[i] = 10 + i;
}

```

```
}

cout << "1. " << *p[1] << endl;

cout << "2. " << int(p[2]) - int(p[1]) << endl;

for (int i=0; i<3; i++)
 t[i] = (t + i) - *(p + i);

cout << "2. " << t[1] << endl;

for (int i=0; i<3; i++)
 t[i] = i;

t[0] = int(p);
p = (int**)(t);

t = (int *) t[0];

cout << "3. " << int(p[2]) << endl;

cout << "4. " << *p[2] << endl;
```