

Examen facultatif, Programmation I

Sciences et Technologies du Vivant, Semestre 1
Chimie et Génie Chimique, Semestre 3

Mercredi 14 décembre 2005

1. N'oubliez pas de mettre vos nom, prénom et SECTION sur toutes vos copies.
2. Vous pouvez rédiger vos réponses au crayon.
3. Les transparents du cours sont les seuls documents autorisés. Vous n'avez droit à aucune autre source d'information, y compris les données et corrigés des exercices.
4. Le nombre de lignes de code entre crochets [] au début des questions est donné à titre indicatif. Vous n'êtes pas obligés de fournir une réponse comportant *exactement* le même nombre de lignes. Par contre, si votre réponse est *beaucoup plus longue* que celle du corrigé, il est certainement possible de faire plus simple.
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. Les questions 6b et 6c sont des questions bonus (au-delà des 100 points). Merci de n'y répondre que si vous êtes à peu près sûr de vous.

Question 1 – Manipulation de tableaux 15 points

On suppose avoir déclaré un tableau d'entiers T

```
const int nb_elements = 10;  
int T[nb_elements];
```

dont les valeurs ont été initialisées.

- (a) [de 4 à 7 lignes] Ecrivez le code qui déclare et initialise un tableau d'entiers S, dont les éléments S[i] devront contenir la somme des éléments de T entre T[0] et T[i] (T[i] compris). (7)

$$S[i] = \sum_{j=0}^i T[j]$$

- (b) [6 lignes] Ecrivez le code qui déclare et initialise un tableau d'entiers P, dont les éléments P[i] devront contenir le nombre d'éléments pairs de T entre T[0] et T[i] (T[i] compris). (8)

Question 2 – Tableaux de booléens 10 points

On suppose avoir déclaré un tableau de booléens T

```
const int nb_elements = 10;  
bool T[nb_elements];
```

dont les valeurs ont été initialisées.

- (a) [2 lignes] Ecrivez le code modifiant ce tableau de manière à remplacer les valeurs `true` par `false` et vice-versa. (5)
- (b) [de 4 à 7 lignes] Ecrivez le code qui affiche `Au moins 1 des elements est vrai` le cas échéant. Le message ne doit être affiché qu'une seule fois au maximum. (5)

Question 3 – Puissances 15 points

- (a) [6 lignes] Ecrivez une fonction qui élève un nombre réel a à une puissance entière n , et retourne le résultat. N'utilisez pas de fonction récursive, ni, bien sûr, la fonction `pow` de `<math.h>`. (7)
- (b) [11 lignes] Ecrivez cette fois la fonction sous forme récursive. N'utilisez pas de boucles `for` ou `while`, ni la fonction `pow`. Pour cela, on remarquera que : (8)

$$a^n = \begin{cases} a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & \text{si } n \text{ est pair et supérieur strictement à } 0 \\ a^{\frac{n-1}{2}} \cdot a^{\frac{n-1}{2}} \cdot a & \text{si } n \text{ est impair} \\ 1 & \text{si } n = 0 \end{cases}$$

Question 4 – Figures graphiques 25 points

Dans cet exercice, vous allez écrire du code permettant d'afficher des figures géométriques à l'aide de caractères `*`.

- (a) [8 lignes] Commencez par écrire une fonction `affiche_ligne`, prenant comme paramètres la largeur totale d'une ligne, ainsi que le nombre d'étoiles à afficher. Les étoiles devront être au centre de la ligne. On suppose pour simplifier que les paramètres sont forcément impairs. Exemple (le caractère `'_'` représente un espace) : (9)

Appel	Résultat
<code>affiche_ligne(5, 3);</code>	<code>_***_</code>
<code>affiche_ligne(7, 3);</code>	<code>__***__</code>

- (b) [3 lignes] Utilisez maintenant la fonction `affiche_ligne` écrite au point précédent afin de dessiner un trapèze isocèle. La nouvelle fonction s'appellera `affiche_trapeze` et prendra comme paramètres les largeurs de la petite et de la grande base du trapèze, ainsi que la largeur totale du dessin. La fonction devra afficher le trapèze au centre du dessin. *iiiiiiii* 1.10 (8)

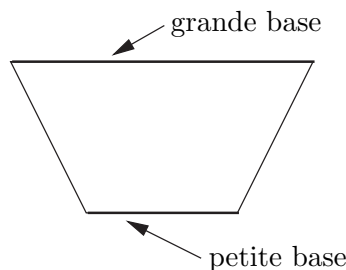


FIG. 1 – Un trapèze isocèle.

Afin de simplifier l'exercice, vous pouvez supposer que les largeurs de la petite base et de la grande base ainsi que la largeur totale sont toujours des nombres impairs.

Exemple : `affiche_trapeze(5, 9, 11);`

```

_*****_
_*****_
_*****_

```

- (c) [4 lignes] Le but de cette troisième partie est d'afficher un sapin similaire à celui vu en exercice, cette fois en utilisant la fonction `affiche_trapeze` du point (b). (8)

```
*
***
*****
***
*****
*****
*****
*****
*****
***
```

Question 5 – Triangle de Pascal 15 points

[de 10 à 20 lignes] Ecrivez une fonction qui calcule et affiche le *Triangle de Pascal*. Cette fonction prendra le nombre de lignes à afficher comme unique paramètre. **Début du Triangle de Pascal :**

```
n = 0: 1
n = 1: 1 1
n = 2: 1 2 1
n = 3: 1 3 3 1
n = 4: 1 4 6 4 1
```

Construction du Triangle On remarquera qu'une ligne n du triangle peut se calculer à partir de la ligne $n - 1$:

- le premier élément et le dernier élément valent 1 ;
- les autres valeurs peuvent se calculer en utilisant la relation

$$p_{i,j} = p_{i-1,j} + p_{i-1,j-1}$$

où $p_{i,j}$ est l'élément sur la ligne i et la colonne j .

La première ligne (pour $n = 0$) est composée d'un 1.

Il est plus simple de ne pas utiliser de fonction récursive. Vous pouvez utiliser deux tableaux, ou mieux, un seul. Ne calculez pas de factorielle ni de coefficient binomial.

Question 6 Mémoire et variables 20 (+ 20 de bonus) points

Indiquez ce qu'affiche le code pour chacune des questions suivantes. Les questions (b) et (c) sont des questions bonus (au-delà des 100 points). Merci de n'y répondre que si vous êtes à peu près sûr de vous.

- (a) **Code :** (20)

```
int a = 5, b = 8;
int * p, * q;

p = &a;
cout << "a1) " << a << " " << *p << endl;
q = p;
*q = b;
cout << "a2) " << *q << endl;
p = q;
*p = a;
cout << "a3) " << *q << endl;
p = &b;
```

```

a = *p;
p = p + 1;
cout << "a4) " << *q << endl;

```

(b) **Code (†) :**

(bonus : 10)

```

void fonction(int *p1, int p2) {
    *p1 = 2 * p2;
}

int main(int argc, char ** argv) {
    int a = 5, b = 6;
    int T[2] = {3, 4};
    int * p = 0;
    int * q = 0;

    fonction(&a, b);
    cout << "b1) " << a << ", " << b << endl;
    p = &a;
    q = &b;
    fonction(q, *p);
    cout << "b2) " << *p << ", " << *q << endl;
    fonction(T + 1, *T);
    cout << "b3) " << T[0] << ", " << T[1] << endl;
    fonction((int *)&p, b);
    cout << "b4) " << a << ", " << b << endl;
    return 0;
}

```

(c) **Code (†) :**

(bonus : 10+une bière)

```

int i = 9;
int * p = &i;
p++;
cout << "c1) " << i << endl;
int j = 14;
int * q = &j;
p = &i;
*p = int(q);
*q = int(p);
p = (int *)(*p);
q = (int *)(*q);
*p = 10;
*q = 20;
cout << "c2) " << j << endl;
p = 0;
cout << "c3) " << i << endl;
cout << "c4) " << *p << endl;

```