

# Examen de Programmation I - bis

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

Lundi 27 Février 2006

1. N'oubliez pas de mettre vos nom, prénom et SECTION sur toutes vos copies.
2. Ne rendez pas la donnée.
3. Vous pouvez rédiger vos réponses au crayon.
4. 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.
5. Le nombre de lignes de code entre crochets [ ] au début des questions est donné à titre indicatif. Dans le cas des fonctions, les en-têtes et les accolades { et } ne sont pas comptées. 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.
6. Les réponses qui font ce qui est demandé mais qui sont inutilement compliquées n'obtiendront pas le maximum des points.
7. 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.
8. Vous pouvez répondre aux questions dans l'ordre de votre choix, à condition de mettre clairement en évidence la référence de la question.

## Question 1 – Mini-Tétris 3D et Graphisme .....45 points

On souhaite représenter une caisse plus ou moins remplie de cubes régulièrement répartis par un tableau à trois dimensions de booléens :

```
bool caisse[LARGEUR] [LONGUEUR] [HAUTEUR] ;
```

`caisse[i] [j] [k]` vaudra `true` si et seulement si il y a un cube à la hauteur `k` en l'emplacement `i`, `j`.

(a) [7 lignes] Ecrivez la fonction (6)

```
int nombre_de_cubes(bool caisse[LARGEUR] [LONGUEUR] [HAUTEUR])  
qui renvoie le nombre de cubes présents dans la caisse.
```

(b) [5 lignes] On veut écrire une fonction permettant d'ajouter un cube en le laissant (10)

tomber d'au dessus l'emplacement repéré par `i`, `j`. Le cube sera évidemment arrêté par les cubes situés en dessous de lui s'il y en a. Ecrivez la fonction

```
void chute(bool caisse[LARGEUR] [LONGUEUR] [HAUTEUR], int i, int j)
```

qui ajoute un cube à `caisse` en le laissant tomber de l'emplacement `i`, `j`.

(c) [6 lignes] Les cubes forment donc des piles dans la caisse. Écrivez la fonction (10)

```
int hauteur_maximale(bool caisse[LARGEUR] [LONGUEUR] [HAUTEUR])  
qui renvoie la hauteur de la plus grande pile de la caisse.
```

(d) [7 lignes] On souhaite représenter graphiquement la caisse vue du dessus. À chaque (15)

emplacement, correspondra un carré gris de 20 sur 20 pixels. Le niveau de gris indiquera la hauteur de la pile : noir pour une pile vide, blanc pour une pile de hauteur `HAUTEUR`, un niveau intermédiaire indiquera une hauteur intermédiaire, calculé linéairement.

Écrivez la fonction

```
void affiche(SimpleWindow * w, bool caisse[LARGEUR] [LONGUEUR] [HAUTEUR])
qui dessine la caisse ainsi dans la fenêtre w, qu'on supposera définie correctement par
ailleurs. On pourra utiliser la fonction (qui existe déjà, il n'y a pas à l'écrire)
void fill_rectangle(int x, int y, int width, int height)
qui dessine un rectangle plein de coin supérieur gauche en x, y, et de dimensions width
et height.
```

## Question 2 – Chatterie ..... 25 points

Une chatterie souhaite organiser son élevage de chats par portée. Pour des raisons de pedigree, on souhaite mémoriser pour chaque portée le père et la mère, ainsi que la date de naissance. Pour chaque chat, on mémorisera également si il est déjà vacciné ou non.

On définira donc les structures suivantes :

```
struct Date
{
    int jour, mois, annee;
};

struct Chat
{
    char * nom;
    char sexe; // 'F' ou 'M'
    bool vaccine;
};

struct Portee
{
    Chat * pere, * mere;
    Chat * chatons;
    int nb_chats;
    Date date_de_naissance;
};

struct Chatterie
{
    Portee * portees;
    int nb_portees;
};
```

(a) [2 lignes] Écrivez la fonction (6)

```
void genitrices(Chatterie * chatterie)
qui affiche, pour chaque portée de chatterie, le nom de la mère.
```

(b) [7 lignes] Écrivez la fonction (6)

```
void a_vacciner(Chatterie * chatterie, Date * date_aujourd'hui)
qui affiche le nom des chatons à vacciner. Un chat doit être vacciné si il ne l'est pas
encore, et si il est agé de plus de 120 jours. Pour connaître l'âge d'un chat, on supposera
qu'on peut utiliser la fonction (on ne demande pas de l'écrire) :
```

```
int nb_jours(Date * date1, Date * date2)
qui renvoie le nombre de jours entre les deux dates passées en paramètre.
```

(c) [8 lignes] Comme la chatterie a beaucoup de chats, elle souhaite avoir une fonction (7)  
permettant de créer automatiquement de nouveaux noms de chat. Écrivez la fonction

```
char * cree_nom(char ** syllabes1, int nb_syllabes1,
               char ** syllabes2, int nb_syllabes2,
               char ** syllabes3, int nb_syllabes3)
```

qui alloue, crée et retourne une chaîne de caractères contenant le nom créé. La première syllabe du nom sera tirée au hasard dans le tableau `syllabes1`, qui est un tableau de `nb_syllabes1` chaînes de caractères. La procédure sera bien sûr la même pour la deuxième et troisième syllabe. Attention, les syllabes n'ont pas forcément toutes le même nombre de caractères.

(d) [5] Écrivez la fonction

(6)

```
Chat * nouveau_chat(char sexe,
                   char ** syllabes1, int nb_syllabes1,
                   char ** syllabes2, int nb_syllabes2,
                   char ** syllabes3_males, int nb_syllabes3_males,
                   char ** syllabes3_femelles, int nb_syllabes3_femelles)
```

qui alloue, crée et renvoie une nouvelle structure `Chat`. Le paramètre `sexe` contiendra M ou F selon que le chaton est un mâle ou une femelle. Cette fonction appellera la fonction `cree_nom` pour créer le nom du chaton. Il va de soit que les noms des mâles et des femelles ne se terminent pas de la même façon, et on utilisera les tableaux `syllabes3_males` ou `syllabes3_femelles` selon le sexe du chaton. N'oubliez pas d'initialiser le champs `vaccine`.

### Question 3 – Pointeurs ..... 30 points

(a) Indiquez ce qu'affiche le code suivant quand on l'exécute :

(10)

```
int a, b;
int * p, * q, * r;

a = 1;
b = 2;
p = &a;
q = &b;
*p = *q;
*q = *p;
cout << "a1) " << a << " " << b << " " << *p << endl;

a = 1;
b = 2;
p = &a;
q = &b;
p = q;
q = p;
cout << "a2) " << a << " " << b << " " << *p << endl;

a = 1;
b = 2;
p = &a;
q = &b;
r = p;
*q = *r;
r = q;
*p = *r;
cout << "a3) " << a << " " << b << " " << *p << endl;
```

```

a = 1;
b = 2;
p = &a;
q = &b;
r = p;
p = q;
q = r;
*q = *p + *r;
cout << "a4) " << a << " " << b << " " << *p << endl;

```

```

a = 1;
b = 2;
p = &a;
q = &b;
r = p;
a = *r;
*q = a;
*r = b;
cout << "a5) " << a << " " << b << " " << *p << endl;

```

(b) Indiquez ce qu'affiche le code suivant quand on l'exécute :

(10)

```

void f(int * p, int x)
{
    *p = *p + x;
    x = x + *p;
}

```

```

int g(int * p, int x)
{
    *p = x;
    x = x + *p;

    return x;
}

```

```

int h(int * p, int x)
{
    return g(p, x);
}

```

```

int * ff(int * p, int x)
{
    *p = x;
    return p;
}

```

Dans la fonction main :

```

int a, b;
a = 1;
b = 2;
f(&a, b);
cout << "b1) " << a << " " << b << endl;

```

```

a = 1;
b = 2;
a = g(&a, b);
cout << "b2) " << a << " " << b << endl;

```

```

a = 1;
b = 2;
f(&a, b);
b = h(&a, a);
cout << "b3) " << a << " " << b << endl;

```

```

a = 1;
b = 2;
int * p = ff(&a, b);
*p = 1;
cout << "b4) " << a << " " << b << endl;

```

```

a = 1;
b = 2;
*ff(&b, b) = 1;
cout << "b5) " << a << " " << b << endl;

```

(c) Indiquez ce qu'affiche le code suivant quand on l'exécute :

(10)

```

int T[10];
int * p;

for(int i = 0; i < 10; i++)
    T[i] = i;

p = T;
cout << "c1) " << p[2] << endl;

p++;
cout << "c2) " << p[2] << endl;

p = &(T[T[T[0] + 1] + 1])

cout << "c3) " << *p << endl;

p = &(T[*T + *(T + 1) + *(T + 2)]);
cout << "c4) " << *p << endl;

for(int i = 0; i < 9; i++)
    T[i + 1] = T[i];
p = T + 9
cout << "c5) " << *p << endl;

```