

Examen intermédiaire, Programmation I

Sciences et Technologies du Vivant, Semestre 1

Mercredi 12 novembre 2008

1. N'oubliez pas de mettre vos NOM et PRÉNOM sur toutes vos copies.
2. Merci de garder la donnée.
3. Vous pouvez rédiger vos réponses au crayon. Cependant, prenez soin d'écrire proprement. Les copies illisibles ne recevront pas de points.
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. Pour l'exercice 3, 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.
6. Les réponses qui font ce qui est demandé mais qui sont trop 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. La question 3.g est une question bonus (au-delà des 100 points).

Question 1 – Commandes Unix 20 points

- (a) On suppose que votre nom d'utilisateur est `username` et que vous appartenez au groupe `sv-ba1`. Vous aimeriez modifier le fichier `exercice.cpp`, à l'aide d'Emacs par exemple. La commande `ls -l`, invoquée sur le répertoire contenant ce fichier, donne le résultat suivant :

```
-rwxrw-r-- 1 lepetit sv-ba1 936 Dec 3 2007 exercice.cpp
```

Est-ce que vous possédez les droits nécessaires pour modifier `exercice.cpp`? Si oui, expliquez pourquoi. Si non, donnez la ou les commandes vous permettant d'obtenir les droits requis pour la modification du fichier.

- (b) Vous vous trouvez dans un répertoire qui contient une longue série de fichiers, dont voici une partie du listing :

```
-rw-r--r-- 1 username sv-ba1 197662 Dec 3 2007 image-0.png
-rw-r--r-- 1 username sv-ba1 197362 Dec 3 2007 image-2.png
-rw-r--r-- 1 username sv-ba1 197866 Dec 3 2007 image-3.png
...
-rw-r--r-- 1 username sv-ba1 197525 Dec 3 2007 image-45.png
-rw-r--r-- 1 username sv-ba1 197398 Dec 3 2007 image-46.png
...
-rw-r--r-- 1 username sv-ba1 197057 Dec 3 2007 image-117.png
-rw-r--r-- 1 username sv-ba1 197951 Dec 3 2007 image-118.png
...
-rw-r--r-- 1 username sv-ba1 197418 Dec 3 2007 image-338.png
```

```

-rw-r--r-- 1 username sv-ba1 196803 Dec  3  2007 image-339.png
...
-rw-r--r-- 1 username sv-ba1 197300 Dec  3  2007 image-2882.png
-rw-r--r-- 1 username sv-ba1 197264 Dec  3  2007 image-2883.png
...
-rw-r--r-- 1 username sv-ba1 197528 Dec  3  2007 image-13429.png

```

Vous aimeriez déplacer tous les fichiers image numérotés de 100 à 999 y compris, dans votre *home directory*. Faites-le en **une seule** commande.

- (c) Vous vous trouvez dans le même répertoire qu'au point précédent et aimeriez supprimer tous les fichiers numérotés de 1000 à 1999. Entrez la commande permettant de réaliser cette opération. (4)
- (d) Toujours depuis le même répertoire, vous aimeriez copier le fichier `image-0.png` dans le répertoire parent du répertoire courant (c'est-à-dire le répertoire supérieur dans l'arborescence du système de fichiers), et le renommer `image.png`. Effectuez ces deux opérations en **une seule** commande. (4)
- (e) On fait un listing d'un répertoire quelconque à l'aide de la commande `ls -la`, et on obtient le résultat suivant : (4)

```

drwxr-xr-x 7 username sv-ba1      4096 2008-10-28 09:25 figs
drwxr-xr-x 2 username sv-ba1      4096 2008-09-26 13:26 .fontconfig
drwxr-xr-x 2 username sv-ba1      4096 2008-09-24 10:29 .fonts
-rw-r--r-- 1 username sv-ba1         44 2008-10-22 16:59 .fonts.cache-1
-r----- 1 username sv-ba1       514 2008-09-24 10:20 .fonts.conf
drwx----- 5 username sv-ba1     4096 2008-11-03 08:45 .gconf
drwx----- 2 username sv-ba1     4096 2008-11-06 16:09 .gconfd

```

Parmi les fichiers retournés par la commande `ls`, y a-t-il des fichiers cachés ? Si oui, lesquels ?

Question 2 – Pointeurs 33 points

- (a) Qu'affiche le programme suivant ? (8)

```
int T[6] = {1, 2, 0, -1, 4, -5};
```

```

int * a = T;
a = a + *a;
cout << "a) " << T[0] << " " << *a << endl;
a = a + *a;
cout << "b) " << T[0] << " " << *a << endl;
a = a + *a;
cout << "c) " << T[0] << " " << *a << endl;
a = a + *a;
cout << "d) " << T[0] << " " << *a << endl;

```

- (b) Qu'affiche le programme suivant ? (9)

```

void f(int a, int * b1, int * b2)
{
    *b1 = +2 * a;
    *b2 = -2 * a;
    a = 0;
}

```

...

```

int a = 1, b, c;
int T[6] = {1, 2, 0, -1, 4, -5};

f(a, &b, &c);
cout << "a) " << a << " " << b << " " << c << endl;

f(T[0], T + 1, T + 2);
cout << "b) " << T[0] << " " << T[1] << " " << T[2] << endl;

f(*T + 1, T + 1, T + 2);
cout << "c) " << T[0] << " " << T[1] << " " << T[2] << endl;

```

(c) Qu'affiche le programme suivant ?

(6)

```

char *fonction(char *string, int n) {
    int l = strlen(string) * n + 1;
    char *newstring = new char[l];
    for (int i=0; i<n; i++)
        strcpy(newstring + i * strlen(string), string);
    return newstring;
}

```

```

int main(int argc, char **argv) {
    char *chaine = "bonjour";
    char *chaine2 = fonction(chaine, 3);

    cout << chaine << " " << chaine2 << endl;
    delete[] chaine2;

    return 0;
}

```

(d) Qu'affiche le programme suivant ?

(10)

```

char c = 'd';
char *p = &c;

cout << "1) " << c << " " << *p << endl;

int *q = new int[5];
for (int i=0; i<5; i++)
    *(q+i) = i % 3;

cout << "2) " << q[2] << " " << q[4] << endl;

for (int i=0; i<5; i++)
    q[i] = 5;

int j = 3;
int *r = q + j;
*r = j;

cout << "3) ";
for (int i=0; i<5; i++)
    cout << q[i] << " ";

```

```

cout << endl;

*q = int(p);
j = q[0] - int(&c);

cout << "4) " << j << endl;

j = q[1];
delete[] q;
*p = *p + 1;

cout << "5) " << j << " " << c << endl;

```

Question 3 – Polynômes 52 points

On souhaite pouvoir effectuer différentes opérations sur des polynômes dans un programme. Pour cela, nous allons représenter les polynômes par des tableaux contenant les coefficients du polynôme. La taille de ces tableaux sera définie par la constante :

```
const int degre_max = 5;
```

ce qui permettra, par exemple, de déclarer un tableau qui contiendra les coefficients d'un polynôme ainsi :

```
float P1[degre_max];
```

L'élément $P1[i]$ contiendra le coefficient du monôme x^i . Par exemple, si les éléments du tableau précédent $P1$ sont initialisés ainsi :

```

P1[0] = -1;
P1[1] = 1;
P1[2] = 0;
P1[3] = 2;
P1[4] = 0;

```

le tableau $P1$ correspond au polynôme $2x^3 + x - 1$.

- (a) [2 lignes] Écrivez la fonction d'en-tête (6)

```
void additionne_polynomes(float * P1, float * P2, float * R)
```

qui permet de calculer la somme de deux polynômes $P1$ et $P2$, passés en paramètre. Le résultat devra être stocké dans le polynôme R . On suppose que ce polynôme R est déjà alloué correctement.

- (b) [4 lignes] Écrivez la fonction d'en-tête (12)

```
float evalue_polynome(float * P, float a)
```

qui permet d'évaluer le polynôme en a , c'est-à-dire qui retourne la valeur

$$\sum_{i=0}^{\text{degre_max}-1} P[i]a^i$$

Pour cela, on pourra utiliser l'algorithme de Hörner, qui est basé sur la factorisation suivante de P :

$$\sum_{i=0}^{\text{degre_max}-1} P[i]a^i = P[0] + a(P[1] + a(P[2] + \dots))$$

La valeur de P en a peut donc être calculée grâce à la suite récurrente définie par :

$$\begin{cases} u_0 = P[\text{degre_max} - 1] \\ u_n = P[\text{degre_max} - n - 1] + au_{n-1} \end{cases}$$

La valeur de P en a est alors donnée par le terme $u_{\text{degre_max}-1}$.

- (c) [1 ligne] Écrivez la fonction d'en-tête (5)

```
float * cree_polynome_nul(void)
```

qui permet de créer un tableau correspondant au polynôme nul, c'est-à-dire le polynôme dont tous les coefficients sont à 0. La fonction devra allouer la mémoire nécessaire, initialiser correctement les coefficients, et retourner un pointeur sur cette mémoire allouée.

- (d) [5 lignes] Écrivez la fonction d'en-tête (6)

```
float * cree_polynome_de_degre_2(float a, float b, float c)
```

qui permet de créer un tableau correspondant au polynôme $ax^2 + bx + c$. Cette fonction `cree_polynome_de_degre_2` devra utiliser la fonction `cree_polynome_nul`.

- (e) [1 ligne] Écrivez la fonction d'en-tête (6)

```
void calcule_racines(float * P, float * racine1, float * racine2)
```

qui suppose que P est un polynôme de degré 2, et qui permet de trouver les racines de l'équation $P = 0$. On suppose qu'on dispose de la fonction

```
void calcule_racines_abc(float a, float b, float c,  
                        float * racine1, float * racine2)
```

qui calcule les 2 racines de l'équation $ax^2 + bx + c = 0$. La fonction `calcule_racines` devra utiliser cette fonction `calcule_racines_abc`.

- (f) [13 lignes] Écrivez la fonction d'en-tête (12)

```
void affiche_polynome(float * P)
```

qui permet d'afficher le polynôme passé en paramètre. Pour le polynôme P_1 de l'exemple ci-dessus, la fonction devra afficher

```
2 x^3 + x - 1
```

La fonction ne devra donc pas afficher les termes dont le coefficient est égal à 0. Attention également à l'ordre de l'affichage.

- (g) [2 lignes] Écrivez la fonction d'en-tête (5 - BONUS)

```
void multiplie_polynomes(float * P1, float * P2, float * R)
```

qui permet de calculer le produit de deux polynômes P_1 et P_2 , passés en paramètre. Le résultat devra être stocké dans le polynôme R . On suppose que ce polynôme R est déjà alloué correctement.

Comme les degrés des polynômes P_1 et P_2 peuvent aller jusqu'à $\text{degre_max} - 1$, le degré de leur produit R peut atteindre $(\text{degre_max} - 1)^2$. Pour simplifier, on ne se souciera pas de ce problème et on supposera que le degré de R ne dépasse jamais $\text{degre_max} - 1$.