

Examen de Programmation I

Sciences et Technologies du Vivant, Semestre 1

Mercredi 17 décembre 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 2, 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. L'examen comporte un maximum de 115 points. La note sera calculée sur 100 points.

Question 1 – Compréhension de code 42 points

(a) Qu'affiche le programme suivant ?

(24 points)

```
#include <iostream>
using namespace std;

struct M
{
    M * g;
    int val;
};

void set(M * m, int v)
{
    m->val = v;
    v = 0;
}

void set2(M * m, M * g2, int v)
{
    m->g = g2;
    g2->val = v;
    g2->g = 0;
}
```

```

void pr(M * m)
{
    if (m->g != 0)
        pr(m->g);
    cout << m->val << " ";
}

int main(int argc, char ** argv)
{
    M a;
    int b = 1;
    set(&a, b);
    cout << "A) " << a.val << " " << b << endl;

    M c;
    set2(&a, &c, 2);
    cout << "B) " << a.val << " " << c.val << endl;

    cout << "C) " << a.g->val << endl;

    M * d = &a;
    cout << "D) " << d->val << " ";
    d = d->g;
    cout << d->val << endl;

    int * e = &(a.val);
    *e = c.val;
    e = 0;
    c.val = 7;
    cout << "E) " << a.val << " " << c.val << endl;

    set(&a, 1);
    set2(&a, &c, 2);
    M f;
    set2(&c, &f, 5);
    cout << "F) ";
    pr(&a);
    cout << endl;

    set(&a, 1);
    set2(&a, &c, 2);
    set2(&c, &f, 5);

    f.g = &a;
    M * h = f.g;
    for(int i = 0; i < 99; i++)
        h = h->g;
    cout << "G) " << h->val << endl;

    h = &a;
    for(int i = 0; i < 100; i++)

```

```

    h = h->g;
    cout << "H) " << h->val << endl;

    return 0;
}

```

(b) Qu'affiche le programme suivant ?

(18 points)

```

#include <iostream>
using namespace std;

struct N
{
    N * g, * d;
    char symbole;
};

N * cree_N(char s, N * g, N * d)
{
    N * n = new N;
    n->symbole = s;
    n->g = g;
    n->d = d;
    return n;
}

int ev(N * n)
{
    if (n->symbole == '+')
        return ev(n->g) + ev(n->d);
    else if (n->symbole == '*')
        return ev(n->g) * ev(n->d);
    else
        // Rappel: int('3' - '0') vaut 3
        return int(n->symbole - '0');
}

int main(int argc, char ** argv)
{
    N * a = cree_N('1', 0, 0);
    N * b = cree_N('3', 0, 0);
    N * c = cree_N('5', 0, 0);
    N * d = cree_N('+', a, b);
    N * e = cree_N('*', d, c);

    cout << "A) " << a->symbole << " " << d->symbole << " " << b->symbole << endl;

    // Rappel: int('3' - '0') vaut 3
    cout << "B) " << int(a->symbole - '0') + int(b->symbole - '0') << endl;

    cout << "C) " << ev(a) << endl;
    cout << "D) " << ev(d) << endl;
    cout << "E) " << ev(e) << endl;
    e->symbole = '+';
}

```

```

    cout << "F) " << ev(e) << endl;

    return 0;
}

```

(c) Qu'affiche le programme suivant ?

(10 points (bonus))

```

char a = 'b';
char c[123] = "d = d + 10;";
char * d = c;
*d = *d + 10;
cout << "A) " << d[0] << endl;
*(d + 1) = *d + 1;
cout << "B) " << d[1] << endl;
d[2] = a + 3;
cout << "C) " << d[2] << endl;
d = d + 3;
*d = 3**d + 3*4;
cout << "D) " << d[3] << endl;
d[1] = d[1] - d[1];
cout << "E) " << c << endl;

```

Question 2 – Gestion d'une bibliothèque 58 points

Dans cet exercice, vous allez écrire un programme permettant de gérer une bibliothèque de livres. Le programme possède, dans sa fonction `main`, deux tableaux contenant respectivement la liste de tous les livres disponibles et la liste de tous les membres inscrits à la bibliothèque :

```

Livre livres[nbr_livres];
Membre membres[nbr_membres];

```

où `nbr_livres` et `nbr_membres` sont des constantes globales entières qui ont été initialisées. Deux structures de données ont déjà été définies : `Date`, permettant de représenter des dates

```

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

```

et `Livre`, qui représente un livre de la bibliothèque.

```

struct Livre {
    char *titre;
    char *auteur;
    Membre *emprunteur;
    Date date_limite;
};

```

Le champ `emprunteur` est un pointeur sur le membre (dont vous aurez à définir la structure à la question (b)) ayant emprunté le livre. Il vaut `NULL` (ou 0) lorsque le livre n'est pas prêté. `date_limite` est la date à laquelle le livre doit être retourné à la bibliothèque. Elle n'est utilisée que lorsqu'un livre est effectivement prêté, et sa valeur est simplement ignorée dans le cas contraire.

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

(8)

```

void affiche_livres(Livre *livres)

```

qui affiche la liste de tous les livres que la bibliothèque possède. Vous afficherez un livre par ligne, et pour chaque livre vous ferez suivre le titre du livre par son auteur entre parenthèses.

Exemple d’affichage :

Guerre Et Paix (Leo Tolstoy)
Madame Bovary (Gustave Flaubert)

- (b) [5 lignes] Définissez une structure `Membre`, afin de représenter un membre. Elle se composera de champs pour le nom, l’adresse e-mail et l’âge du membre. Nous supposons, pour simplifier, qu’un membre ne peut emprunter qu’un livre à la fois. Ajoutez donc un quatrième champ, intitulé `livre` et de type pointeur sur `Livre`, qui référence le livre que le membre a emprunté. Ce champ prendra la valeur `NULL` (ou `0`) lorsque le membre n’a pas emprunté de livre. (8)

- (c) [6 lignes] Écrivez la fonction d’en-tête (8)

```
void affiche_membres(Membre *membres)
```

qui affiche la liste de tous les membres de la bibliothèque, ainsi que le titre du livre qu’ils ont emprunté, pour autant qu’ils en aient emprunté un.

Vous prendrez soin d’afficher un membre par ligne en respectant la syntaxe illustrée dans l’exemple suivant :

Sofia Coppola (sofia.coppola@epfl.ch) - Madame Bovary
Ethan Coen (ethan.coen@unil.ch)
Quentin Tarantino (quentin.tarantino@ethz.ch) - Le Rouge Et Le Noir

- (d) [7-12 lignes] Écrivez la fonction d’en-tête (8)

```
void nouveau_membre(Membre *m)
```

qui demande à l’utilisateur de saisir au clavier les données d’un nouveau membre et les stocke dans la structure `Membre` passée en paramètre. On considère que cette structure a déjà été allouée en mémoire. Prenez soin d’initialiser correctement tous les champs de la structure. On suppose, bien entendu, qu’un nouveau membre n’a encore emprunté aucun livre.

- (e) [9 lignes] Dans notre bibliothèque, on aimerait que les titres des livres soient tous écrits de la même manière, avec chaque mot commençant par une majuscule. Vous allez écrire une fonction qui vérifie qu’un titre est bien dans ce format et le modifie au besoin. Pour simplifier, on supposera que les titres des livres ne sont formés que de lettres et d’espaces (pas de chiffres ni de ponctuation). (9)

Votre fonction prendra l’en-tête suivant

```
void capitalise(char *titre)
```

Exemple d’application : *guerre et paix* sera transformé en *Guerre Et Paix*.

- (f) [13 lignes] Écrivez la fonction d’en-tête (8)

```
bool emprunte(Membre *m, Livre *l)
```

qui reçoit en paramètres un pointeur sur `Membre` ainsi qu’un pointeur sur `Livre`. La fonction devra modifier les deux structures, afin qu’elles reflètent le fait que le membre `m` a emprunté le livre `l`. La fonction devra notamment vérifier que le membre peut emprunter un livre (c’est-à-dire qu’il n’en a pas déjà emprunté un) et que le livre n’est pas déjà prêté à un autre membre. On suppose qu’un livre est toujours prêté pour une durée d’un mois.

La fonction retourne `true` si l’opération s’est bien déroulée, et `false` si le prêt n’a pas pu être effectué.

Pour écrire cette fonction, vous pouvez faire appel à la fonction suivante

```
Date date_courante()
```

qui retourne la date du jour.

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

```
int difference_dates(Date *d1, Date *d2)
```

qui calcule le nombre de jours de différence entre deux dates. Pour simplifier, on suppose que chaque mois est composé de 30 jours, et donc qu'une année ne compte que 360 jours. Le résultat sera positif, si `d1` est plus récente que `d2`, et négatif dans le cas contraire. Par exemple, si `d1` est le 17 décembre 2008 et `d2` le 17 novembre 2008, le résultat sera 30 jours.

- (h) [6 lignes] Écrivez la fonction d'en-tête (9)

```
void verifie_retards(Membre *membres)
```

dont le but est de vérifier si des membres ont oublié de rendre le livre qu'ils ont emprunté, et de les prévenir par e-mail, le cas échéant. Pour ce faire, on suppose qu'on a accès à la fonction suivante

```
void envoie_email_retard(char *adresse, char *titre, int nbr_jours)
```

qui envoie un e-mail de rappel à l'adresse passée en paramètre. La fonction nécessite qu'on lui passe également le titre du livre concerné ainsi que le nombre de jours de retard.