

Examen de Programmation I

Sciences et Technologies du Vivant, Semestre 1
Mercredi 22 décembre 2010

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.
4. Tout document autorisé, machines interdites.

Question 1 – Compréhension de code ... 50 points + 7 points de bonus

Qu'affiche le programme suivant ? Les différentes parties sont la plupart indépendantes les unes des autres.

```
#include <iostream>

using namespace std;

struct E
{
    char v;
};

struct L
{
    E es[3];
};

struct I
{
    L * s;
    I * i;
};

void F(E * e)
{
    cout << e->v;
}

void G(L * l)
{
    for(int i = 0; i < 3; i++) {
        F(l->es + i);
    }
    cout << endl;
}
```

```

void G(I * i)
{
    if (i != 0) {
        G( i->s );
        G( i->i );
    }
}

void K(char c1, char c2)
{
    char c = c1;
    c1 = c2;
    c2 = c;
}

int main(int argc, char ** argv)
{
    E a, b, c;
    a.v = ':';
    b.v = '-';
    c.v = ')';

    cout << "A) "; // 3 points
    F(&a); F(&b); F(&c);
    cout << endl;

    L d;
    d.es[0] = a;
    d.es[1] = b;
    d.es[2] = c;
    c.v = '(';

    cout << "B) "; // 3 points
    G(&d);

    E * e = d.es;
    e++;
    e[1].v = 'X';

    cout << "C) "; // 3 points
    G(&d);

    E * f = e - 1;
    f->v = 'v';
    e[0].v = 'd';

    cout << "D) "; // 4 points
    G(&d);
}

```

```

d.es[0].v = ':';
d.es[1].v = '-';
d.es[2].v = ')';

K(d.es[0].v, d.es[2].v);

cout << "E) "; // 5 points
G(&d);

d.es[0].v = ':';
d.es[1].v = '-';
d.es[2].v = ')';

for(int i = 0; i < 2; i++) {
    d.es[i] = d.es[i+1];
}

cout << "F) "; // 5 points
G(&d);

d.es[0].v = ':';
d.es[1].v = '-';
d.es[2].v = ')';

for(int i = 0; i < 2; i++) {
    d.es[i+1] = d.es[i];
}

cout << "G) "; // 5 points
G(&d);

d.es[0].v = ':';
d.es[1].v = '-';
d.es[2].v = ')';

I g;
g.s = &d;
g.i = 0;

L h;
h.es[0].v = 'X';
h.es[1].v = 'D';
h.es[2].v = ' ';

I i;
i.s = &h;
i.i = &g;

cout << "H) "; // 5 points
G(&g);

```

```

cout << "I) "; // 5 points
G(&i);

char * as[3];
for(int i = 0; i < 3; i++)
    as[i] = &(h.es[i].v);

*(as[0]) = '8';
as[1] = as[0];
*(as[2]) = *(as[1]);

cout << "J) "; // 6 points
G(&h);

h = d;
*(as[2]) = '(';
E * m = d.es;
m = &(h.es[2]);
cout << "K)"; // 6 points
G(&i);

I * j = &i, * k = j->i;
k->i = j;
cout << "L)"; // 7 points (bonus)
G(j->i->i);
cout << "lol" << endl;

return 0;
}

```

Question 2 – Poker 50 points + 8 points de bonus

Le but de cet exercice est d'écrire plusieurs fonctions d'un programme permettant de jouer au poker. Les questions sont indépendantes entre elles. Vous pouvez répondre à une question même si vous n'avez pas su faire les précédentes.

Le nombre de lignes est donné à titre indicatif. Il tient compte de la ligne d'en-tête de la fonction, mais pas des lignes vides ou qui ne contiennent que des accolades { ou }.

- (a) [5 lignes] Le poker se joue avec un jeu de 52 cartes. Pour représenter une carte, nous allons utiliser la structure suivante : (7 points)

```

struct Carte {
    int couleur;
    int niveau;
};

```

Le champs `couleur` contiendra une valeur entre 0 et 3, correspondant à l'une des couleurs (cœur, carreau, trèfle, pique). Le champs `niveau` contiendra une valeur entre 2 et 14 (11 pour le valet, 12 pour la dame, 13 pour le roi, et 14 pour l'as).

Nous allons représenter le jeu de cartes par un tableau de 52 cartes :

```

Carte jeu[52];

```

Ecrivez la fonction `initialiser_jeu` qu'on pourra appeler ainsi :

```

    initialiser_jeu(jeu);

```

Après cet appel :

- le champ	couleur	de l'élément	0	de jeu devra contenir	0;
- le champ	niveau	de l'élément	0	de jeu devra contenir	2;
- le champ	couleur	de l'élément	1	de jeu devra contenir	0;
- le champ	niveau	de l'élément	1	de jeu devra contenir	3;
- ...					
- le champ	couleur	de l'élément	12	de jeu devra contenir	0;
- le champ	niveau	de l'élément	12	de jeu devra contenir	14;
- le champ	couleur	de l'élément	13	de jeu devra contenir	1;
- le champ	niveau	de l'élément	13	de jeu devra contenir	2;
- ...					
- le champ	couleur	de l'élément	51	de jeu devra contenir	3;
- le champ	niveau	de l'élément	51	de jeu devra contenir	14.

- (b) [8 lignes] Ecrivez la fonction `melanger_jeu` qui permet de mélanger le jeu de cartes en appelant : (7 points)

```
melanger_jeu(jeu);
```

Une façon de mélanger les éléments d'un tableau est de tirer deux indices au hasard, échanger les éléments stockés à ces deux indices, et de répéter plusieurs fois cette opération.

- (c) [4 lignes] Pour simplifier, nous allons supposer que chaque joueur reçoit 5 cartes, qui constituent une *main*. Pour pouvoir représenter une main, nous allons utiliser la structure suivante : (7 points)

```
struct Main {
    Carte cartes[5];
};
```

Nous allons également supposer qu'il n'y a que deux joueurs, dont les mains seront déclarées de la façon suivante :

```
Main main_joueur1, main_joueur2;
```

Ecrivez la fonction `distribuer` qu'on pourra appeler de la façon suivante :

```
distribuer(jeu, &main_joueur1, &main_joueur2);
```

Après cet appel :

- l'élément d'indice	0	du champ <code>cartes</code> de	<code>main_joueur1</code>	devra contenir la même carte que
l'élément d'indice	0	du tableau <code>jeu</code> ;		
- l'élément d'indice	0	du champ <code>cartes</code> de	<code>main_joueur2</code>	devra contenir la même carte que
l'élément d'indice	1	du tableau <code>jeu</code> ;		
- l'élément d'indice	1	du champ <code>cartes</code> de	<code>main_joueur1</code>	devra contenir la même carte que
l'élément d'indice	2	du tableau <code>jeu</code> ;		
- ...				
- l'élément d'indice	4	du champ <code>cartes</code> de	<code>main_joueur2</code>	devra contenir la même carte que
l'élément d'indice	9	du tableau <code>jeu</code> .		

- (d) [3 lignes] On suppose qu'on a déjà écrit la fonction d'en-tête (7 points)

```
void affiche_carte(Carte * c)
```

qui affiche la carte (dans la fenêtre Terminal, par exemple `Dame de coeur`) dont l'adresse est passée en paramètre.

Ecrivez la fonction d'en-tête :

```
void affiche_main(Main * m)
```

qui affiche les cartes de la main dont l'adresse est passée en paramètre, en appelant la fonction `affiche_carte`.

- (e) [5 lignes] Nous allons maintenant passer aux fonctions qui permettent d'évaluer une main. Une main est appelée une "flush" si les 5 cartes qu'elle contient sont de la même couleur. (7 points)

Ecrivez la fonction d'en-tête

```
bool est_ce_une_flush(Main * m)
```

qui renverra `true` si la main dont l'adresse est passée en paramètre est une flush, `false` sinon.

- (f) [5 lignes] Pour simplifier l'écriture des fonctions permettant de trouver les autres types de mains, nous allons ajouter un champ à la structure `Main`, qui devient : (7 points)

```
struct Main {  
    Carte cartes[5];  
    int compteurs[15];  
};
```

Le tableau `compteurs` permettra de savoir combien la main contient de cartes de même niveau. Par exemple, si une main contient deux valets, l'élément 11 de son champ `compteurs` devra contenir 2.

Ecrivez donc la fonction d'en-tête :

```
void calcule_compteurs(Main * m)
```

qui calcule la valeur des éléments du tableau `compteurs` de la main dont l'adresse est passée en paramètre. On supposera que les éléments du tableau `cartes` sont déjà correctement initialisés.

- (g) [5 lignes] Une main est un carré si elle contient 4 cartes de même niveau. (8 points)

Ecrivez la fonction d'en-tête

```
bool est_ce_un_carre(Main * m)
```

qui renverra `true` si la main dont l'adresse est passée en paramètre est un carré, `false` sinon.

On supposera qu'on a déjà appelé la fonction `calcule_compteurs`, c'est-à-dire que les éléments du tableau `compteurs` sont déjà correctement initialisés.

- (h) [9 lignes] Une main est une suite si elle contient 5 cartes dont les niveaux se suivent, quelque soient leurs couleurs. Par exemple, la main : (8-bonus)

```
4 de coeur  
2 de pique  
6 de coeur  
5 de pique  
3 de trefle
```

est une suite.

Ecrivez la fonction d'en-tête

```
bool est_ce_une_suite(Main * m)
```

qui renverra `true` si la main dont l'adresse est passée en paramètre est une suite, `false` sinon.

Comme pour la question précédente, on supposera qu'on a déjà appelé la fonction `calcule_compteurs`, c'est-à-dire que les éléments du tableau `compteurs` sont déjà correctement initialisés.

La main :

```
As de coeur  
2 de pique  
3 de coeur  
4 de pique  
5 de trefle
```

est également une suite, mais nous ne tiendrons pas compte de ce cas.