Trimestre Automne, 2016

Mohamed Lokbani

IFT1169 – Examen Intra –

Inscrivez tout de suite : votre nom et le code permanent.

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : Vendredi 14 octobre 2016

Durée : 2 heures (de 16h30 à 18h30)

Local : Z-310, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1._____/20 (1.1 à 1.10)

2._____/20 (2.1, 2.2, 2.3)

3._____/20 (3.1, 3.2, 3.3)

4._____/20 (4.1, 4.2)

5._____/20 (5.1)

Total : _____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* Toute infraction relative à une fraude, un plagiat ou un copiage est signalée par le surveillant au directeur de département ou au professeur qui suspend l'évaluation.

F.A.S

Exercice 1 (20 points) trouvez la meilleure correspondance pour les descriptions ci-dessous en y associant un nombre de 1 à 10. Le nombre ne peut-être utilisé qu'une seule fois. Nous avons complété une description à titre d'exemple.

1. Une partie de la classe que tout le monde a le droit d'y accéder.
2. Est appelé quand une instance de la classe cesse d'exister.
3. Elle est utilisée pour récupérer le contenu d'une donnée membre protégée.
4. Une partie de la classe que seulement quelques-uns ont le droit d'y accéder.
5. Elle sert d'interface pour manipuler les données membres.
6. La fonction peut accéder à la partie privée de la classe.
7. Elle définit un algorithme ou une structure indépendante du type de données employé.
8. La variable est stockée au sein d'une classe.
9. Est appelé quand une instance de la classe commence à exister.
10. Elle est utilisée pour modifier le contenu d'une donnée membre protégée.
11. Il permet d'automatiser la compilation d'un projet.

Fonction membre	
Mutateur	
Public	
Donnée membre	
Makefile	11
Template	
Destructeur	
Friend	
Constructeur	
Accesseur	
Private	

Exercice 2 (20 points)

2.1 Commencez par encercler la bonne réponse pour chacune des méthodes liées aux propriétés ci-dessous :

Cas	Propriétés	Fonction getline		L'opérateur >>	
1	Ignore tous les espaces blancs	VRAI	FAUX	VRAI	FAUX
2	Peut écrire une donnée directement dans un int	VRAI	FAUX	VRAI	FAUX
3	Peut planter un stream s'il n'y a plus de données à lire	VRAI	FAUX	VRAI	FAUX
4	Peut-être utilisé avec un ofstream	VRAI	FAUX	VRAI	FAUX

Par la suite, expliquez brièvement votre réponse.

Cas 1 :

Cas 2 :

Cas 3 :

Cas 4 :

2.2 Quelle sera la valeur de la variable « b » (ligne 5) après l'exécution de ce bout de code (**la syntaxe est OK**) :

```
1    istringstream iss("text");
2    string s;
3    iss >> s;
4    bool b = iss.fail();
5    cout << "b: " << b << endl;
```

Expliquer brièvement votre réponse.

2.3 Quelle sera la valeur de « x » (ligne 7) après l'exécution de ce bout de code (**la syntaxe est OK**) :

```
1    ostringstream oss;
2    for (int i = 5; i< 8; i++)
3           oss << i;
4    istringstream iss(oss.str());
5    int x;
6    iss >> x;
7    cout << "x: " << x << endl;
```

Expliquer brièvement votre réponse.

Exercice 3 (20 points)

3.1 Quelle sera la valeur de « A » (ligne 4) après l'exécution de ce bout de code (**la syntaxe est OK**) :

```
1    string A = "Bonjour";
2    string B = A;
3    B[0] = 'X';
4    cout << "A: " << A << endl;
```

Expliquer brièvement votre réponse.

3.2 Quelle sera la valeur de « Z » après l'exécution de ce bout de code (**la syntaxe est OK**) :

```
1    char a[] = "abc";
2    char b[] = "abc";
3    bool Z = (a == b);
4    cout << "Z: " << Z << endl;
```

Expliquer brièvement votre réponse.

3.3 Soit le bout de code suivant :

```
1    int p[2];
2    int* x = new int[3];
3    int* y = new int[5];
4    int* z = y;
5    y = x;
6    x = p;
7    delete z;
8    delete y;
9    delete x;
```

Même si le bout de code compile correctement, il y a quand même quelques bogues. Lesquels? Encerclez la bonne réponse.

Cas	Situation	Le code a ce bogue	
1	Ne peut pas faire « delete z », car elle n'a pas été allouée avec new	VRAI	FAUX
2	Il a une fuite de mémoire	VRAI	FAUX
3	Le fait de mixer des tableaux de tailles différentes	VRAI	FAUX
4	On essaye de libérer un espace mémoire statique	VRAI	FAUX
5	Utilisation d'une mauvaise syntaxe pour libérer l'espace mémoire	VRAI	FAUX

Par la suite, expliquez brièvement votre réponse.

Cas 1 :

Cas 2 :

Cas 3 :

Cas 4 :

Cas 5 :

Exercice 4 (20 points)

4.1 Soit le bout de code suivant (**la syntaxe est OK**) :

```
1  int main(int argc, char* argv[]){
2          int x = atoi(argv[1]); // atoi : string to int
3          int y;
4          cin >> y;
5          cout << x + y << endl;
6
7          return 0;
8  }
```

a) Tout en expliquant brièvement votre réponse, que va afficher en sortie le précédent programme suite à ces commandes (↵ est le retour de chariot) :

```
4_1.exe 10 ↵
18 ↵
```

b) Tout en expliquant brièvement votre réponse, que va afficher en sortie le précédent programme suite à ces commandes (↵ est le retour de chariot) :

```
4_1.exe 10 | 4_1.exe 2 ↵
18 ↵
```

C'est la même chose que d'écrire (`4_1.exe 10 ↵ 18 ↵`). Ceci n'est autre que la question (a). Le résultat de cette opération est stocké en mémoire pour être fourni comme une donnée à (`4_1.exe 2 ↵` « mémoire » ↵). Le système va gérer pour vous cette mémoire temporaire.

4.2 Soit le bout de code suivant (**la syntaxe est OK**) :

```
 1  int main(int argc, char* argv[]){
 2    int pos1 = 0, pos2 = 0, var1 = argc-1, var2, autre2;
 3    cin >> var2 >> autre2;
 4    cout << var1 + var2 << " ";
 5    while (pos1 < var1 || pos2 < var2){
 6         bool test1 = (pos2 == var2);
 7         if (!test1)
 8                test1 = (pos1 < var1) && (atoi(argv[1+pos1]) < autre2);
 9         if (test1) {
10                cout << argv[1+pos1] << " ";
11                pos1++;
12         } else {
13                cout << autre2 << " ";
14                pos2++;
15                if (pos2 < var2)
16                       cin >> autre2;
17         }
18    }
19    cout << endl;
20    return 0;
21  }
```

a) Tout en expliquant brièvement votre réponse, que va afficher en sortie le précédent programme suite à ces commandes (↵ est le retour de chariot) :

```
4_2.exe 3 ↵
1 5 ↵
```

b) Tout en expliquant brièvement votre réponse, que va afficher en sortie le précédent programme suite à ces commandes (↵ est le retour de chariot) :

```
4_2.exe 3 | 4_2.exe 1 2 4 ↵
1 5 ↵
```

Même principe que la question (**4.1.b**). D'abord on exécute (`4_2.exe 3 ↵ 1 5 ↵`). Le résultat est en mémoire, puis on exécute (`4_2.exe 1 2 4 ↵` « mémoire » ↵). Le système va gérer pour vous cette mémoire temporaire.

Exercice 5 (20 points)

On vous demande d'écrire la fonction « C++ » « copierStrTab », dont la signature est : char** copierStrTab(int,char**) Cette fonction accepte donc deux arguments, le premier est un entier qui est la taille du tableau, le second est un tableau de chaînes de caractères. La fonction retourne un pointeur vers **un nouvel espace** contenant le tableau copié. Cette copie est appelée dans le jargon, une copie profonde. Cette fonction peut-être utilisée comme suit :

```
int main(int argc, char** argv){
        char **ptr;
        ptr = copierStrTab(argc,argv);
        // etc.
```

Le pointeur « ptr » va pointer un nouvel espace ayant une copie complète du contenu du tableau « argv ». Vous allez supposer que vous avez assez de mémoire pour contenir tout le contenu du tableau « argv ». Vous devez utiliser les notions « C++ » pour l'allocation de mémoire. Vous ne devez allouer que le strict nécessaire de mémoire afin d'effectuer cette copie.

```
char **copierStrTab(int nombre,char **tableau){




}
```