Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara de Boumerdès, Faculté des Sciences,
Département : Informatique

Examen (2015/2016) : L2-S3	Module : POO	Durée : 01h30	Préparation : Gaceb

Exercice 1 : Questions de cours (7pts)

1) Expliquer la différence entre :
- i++ et ++i
- Paramètres formels et réels d'une fonction.

2) Soit le code suivant : *int *p, n=5;* comment récupérer l'adresse de n dans p.

3) Indiquer les trois modes de transmission de variables à une fonction, expliquer le rôle de chacun.

4) Soient trois tableaux t1 (1D), t2(2D), t3 (3D) de tailles n pour t1, n×m pour t2 et n×m×l pour t3, écrire les lignes de code permettant d'allouer dynamiquement la mémoire pour t1, t2 et t3 et de libérer leurs mémoires allouées.

5) Soient les classes suivantes

```
class A {
 private:    int x;
 public:     int y;
 protected:  int z;
};
class B : public    A{…};
class C : protected, A{…};
class D : private   A{…};
```

Indiquer le niveau d'accès (public, private, protected, ou non accessible) des membres x,y et z de A dans chacune des classes B, C et D.

6) Pour quelle raison on déclare virtuelle une superclasse ? Expliquer en donnant un exemple de 4 classes. Préciser à quel niveau faut-t-il indiquer la déclaration virtuelle.

7) Donner la définition d'une classe abstraite, d'une liaison statique et d'une liaison dynamique.

Exercice 2 : (9 pts)

La classe *Tab* suivante comprend un tableau *data* d'entiers, un constructeur permettant de créer un tableau encapsulé d'une taille *n* transmise en paramètre et d'initialiser ses éléments à *0*, un destructeur, une méthode *put* permettant de placer un entier dans le tableau à un indice donné en paramètre, et une méthode *isIn* qui teste si un entier passé en paramètre est dans le tableau.

```
class Tab {
  private:
  int n;
  int* data;
  public:
  Tab (int n=0);
  void put(int i, int a);
  int isIn(int a);
  ~Tab();
};
```

1) Complétez les définitions de constructeur, destructeur et méthodes *put* et *isIn*.

```
a)Tab::Tab(int n){?}
b)Tab::~Tab(){?}
c)void Tab::put(int i,int a){?}
d)int Tab::isIn(int a){?}
```

2) Ecrire deux fonctions en ligne membres de la classe *Tab* permettant de saisir les données de tableau *data* par l'utilisateur et de les afficher sur l'écran.

3) Toujours en mode en ligne, ajouter la méthode *add* permettant d'afficher la somme élément par élément de tableau *data* de l'objet en cours avec celui d'un objet de même type *Tab* passé en paramètre à cette fonction par valeur. Si les deux tableaux ont une taille différente, un message d'erreur devra être affiché par la méthode.

4) Pour que la fonction *add* fonctionne sans incident, faut-il surcharger la classe *Tab* par un constructeur par recopie ? Si oui expliquer la raison et ajouter sa définition et sa déclaration dans la classe.

5) Définir l'opérateur + pour l'addition de deux objets *T1* et *T2* de types *Tab*. Avec cette opérateur on doit pouvoir écrire *T3=T1+T2*. En cas où *T1* et *T2* n'ont pas la même taille *(T1.n>T2.n)*, *T3* prendra la taille et les données de T1 qui sont sommées aux données de *T2 du 0* jusqu'à *T2.n*.

6) Définir l'opérateur *[ ]* prenant un paramètre *i* et renvoyant la valeur de $i^{ème}$ élément de tableau. Pour un objet *T* de type *Tab*, cet

opérateur doit nous permettre de faire les 2 opérations suivantes : *int v=T[i]* pour affecter la valeur de *T.data[i]* à la variable v, ainsi que *T[i]=v* pour affecter la valeur de *v* à *T.data[i]*.

7) Ajouter une méthode qui trie les éléments de tableau data dans l'ordre croissant.

8) Ajouter une fonction récursive *somme* qui renvoie la somme des éléments de tableau *data* d'un objet donné. Cette fonction ne doit pas utiliser des boucles for, while ni goto.

9) Créer un patron de classe *Tab* en modifiant sa définition et sa déclaration pour pouvoir gérer différents types de données (double, float, int, etc.) dans *data*.

Exercice 3 : 4 pts

1) Ecrire la déclaration et la définition des classes nécessaires au fonctionnement du programme suivant (en ne fournissant que les méthodes nécessaires à ce fonctionnement). N'oubliez pas de commenter votre code :

```
main()
{
// Tableau de trois figures géométriques
Forme ** figures = new Forme*[3];

//Création d'un carré de 2 cm de coté
figures[0] = new Carre(2);

//Création d'un cercle de 3 cm de rayon
figures [1] = new Cercle(3);

//Création d'un rectangle H=5.2 et L=2 cm
figures [2] = new Rectangle(5.2, 2);

//Affichage des surfaces
for (int i=0 ; i<3; i++)
{
  figures[i]->affichetype();
  cout <<":surface=")
      <<figures[i]->getSurface()
      <<"cm2"<<endl;
}

delete [] figures ;
}
```

Sortie de ce programme est :

```
Carré (coté 2.0 cm) : surface = 4.0 cm2
Cercle (rayon 3.0 cm ): surface = 28.26 cm2
Rectangle (H=5.2 cm et L=2) : surface =
10,40 cm2
```

2) Ajouter des destructeurs à toutes les classes affichant à l'écran le nom de la classe de l'objet détruit. Donner le résultat de l'affichage à la fin de l'exécution de programme main ci-dessus.

Rappel :

- Surface d'un carré = coté×coté
- Surface d'un cercle = Pi×rayon×rayon
- Surface d'un rectangle = Hauteur×Largeur

Bon courage