Examen de Programmation Orientée Object en C++

Exercice 1 : Questions de cours (1+0.5x5+1+1+1=6 points)

1. Donner l'extension(s) d'un fichier source C++
2. Définir les concepts suivant : Programmation orientée objet, Objet, Classe, polymorphisme, héritage
3. Donner la signification des mots clés suivant :
   Virtual ; Private.

4. Quelle est la sortie du code suivant? (1.5 points)

```
#include <iostream>
using namespace std;
class calculer {
int x, y;
public:
void val(int, int);
int somme() {
return (x + y);
}
};
void calculer::val(int a, int b) {
x = a;
y = b;
}
int main() {
calculer calculer;
calculer.val(5, 10);
cout << "La somme = " << calculer.somme();
return 0;
}
```

A- La somme = 5
B- La somme = 10
C- La somme = 15
D- Erreur parce que calculer est utilisé comme nom de classe et nom de variable dans la ligne 19.

Exercice 2(choisir la(les) bonne(s) reponse(s)): (0.25x8=2pts)

1. Shape*fig;

En supposant que le code ci-dessus est valide, on dit que fig est_____ de ____Shape.

a)une variable / la classe b)une variable / l'instance c)une instance / la classe

2. Shape*fig = new Square(42);

Ici, on dit que le _____de fig est Square *et que son _____est Shape*.

a)type statique / type dynamique
b) type dynamique / type statique

3. Le code de la question précédente implique que _____dérive de_____.

a)Shape/Square b)Square/Shape

4. Quelle syntaxe permet de modifier la cible d'un pointeur p?
   a)*p = a; b)p = a; c)&p = a; c)impossible
5. Dans une classe, les constructeurs de ses objets membres sont appelées le code de son constructeur

a)Avant b)après c) ¸ca dépend du compilateur

6. Dans une classe, les destructeurs de ses objets membres sont appelés le code de son destructeur.
   a)Avant b)après c) dépend du compilateur
7. En C++, la déclaration d'une classe :
   peut contenir 0 ou plusieurs constructeurs
   a)doit toujours contenir 2 constructeurs b)ne doit contenir qu'1 seul constructeur c)doit contenir au moins 1 constructeur
8. En C++, on distingue des méthodes surchargées en fonction :

a)de leurs noms b)uniquement du nombre de leurs paramètres c) de leurs types de retour d)du nombre ou du type de leurs paramètres

Problème :
(0.75+0.75+0.75+0.75+0.75*4+0.75x2+0.75+0.75+0.75+0.75+0.75x2=12points)

Dans cet problème, nous allons définir une classe Complexe qui permettra de manipuler des nombres complexes et d'implémenter ainsi certaines opérations algébriques standards.

1. Écrire une classe Complexes permettant de représenter des nombres complexes. Un nombre complexe Z comporte une partie réelle et une partie imaginaire : Z = PartieRéelle + PartieImaginaire * i (On notera la parie reelle **re** et la partie imaginaire **im**)
2. Définir un **constructeur** par défaut sans paramètre permettant d'initialiser les deux parties du nombre à 0.
3. Définir un **constructeur** d'initialisation pour la classe.
4. Écrire un dernier constructeur **public Complexe(Complexe c)** permettant de créer une copie du Complexe passé en argument.
5. Écrire des méthodes publiques **public Complexe plus(Complexe c),public Complexe fois(Complexe c),public Complexe divise(Complexe c),et public double module()** qui implémentent les opérations algébriques classiques sur les nombres complexes (la racine carrée est donnée par sqrt(double d)).
6. Ajouter des méthodes **plus**, et **fois** qui prennent des **double** en paramètres.
7. Écrire une méthode **afficher ( )** qui donne une représentation d'un nombre complexe comme suit : a+b*i.
8. Écrire une méthode **bool egal(Complexe c)** permettant de comparer 2 complexes. **utiliser ¨==¨pour faire une comparaison**
9. Ajouter une méthode **toString()** renvoyant une représentation sous forme de chaine de caractère du Complexe courant
10. Écrire une méthode void swap(Complexe c1, Complexe c2) permettant de permuterc1 etc2. Par exemple, on voudrait que le code suivant :
    Complexe c1=new Complexe(1, 1);
    Complexe c2=new Complexe(2,2);
    Complexe.swap(c1,c2)
    Complexe.affiche(c1);
    affiche ¨2+2i¨.
    Pourrait-on écrire une telle méthode pour permuter des entiers ?

11. Écrire des méthodes conjugue() et inverse()qui transforment un complexe en son conjugué ou en son inverse.
    NB : ces méthodes ne retournent rien : elles modifient juste le Complexe sur lequel elles sont appelées.

Une bonne présentation de la copie est conseillée.

`public`: niveau le plus bas de protection, toutes les données ou fonctions membres d'une classe sont utilisables par toutes lesfonctions

`private`: niveau le plus élevé de protection, données (ou fonctionsmembres) d’une classe utilisables uniquement par les fonctionsmembre de la même classe
`protected`: comme `private` avec extension aux classes dérivées