Nom: .....Prénom: ..... الفوج: ..... اللقب: ..... الإسم:

Question 1: Cocher les énoncés corrects (note: 0 énoncé non coché, +1 énoncé correct coché, -1 énoncé incorrect coché) (**10 points**)

1) Une classe représente le moule pour les objets
2) Un objet représente une instance pour une classe
3) La relation d'héritage est une relation entre plusieurs objets
4) dans la relation d'héritage, une classe mère hérite sa classe fille
5) un attribut public est hérité avec la même visibilité dans la classe fille
6) Un TAD est objet
7) Un TAD représente l'implémentation d'un objet
8) Dans la partie sémantique du TAD on met la description algorithmique des opérations
9) Une spécification algébrique contient la partie syntaxique où on déclare les attributs et les méthodes
10) Une classe est la version C++ de ce que nous appelons TAD dans le niveau abstrait
11) Pour éviter l'interdiction d'accès aux attributs privés, il est préférable de les rendre tous public
12) Il est toujours mieux de déclarer des attributs publics que d'utiliser les setters et les getters
13) Un TAD doit contenir des axiomes dans sa spécification algébrique
14) Un axiome représente un invariant toujours correct
15) Un axiome est une condition logique toujours vraie
16) L'héritage est un mécanisme de réutilisation
17) les attributs privés d'une classe ne sont accessibles que dans cette classe et dans le programme main
18) Un TAD générique représente un moule pour les TAD spécifiques
19) Un attribut privé est accessible dans sa classe est dans toutes ses classes filles
20) Le mécanisme d'héritage est pour les classes ce qui est le mécanisme d'enrichissement pour les TADs.
21) Un TAD peut déclarer des attributs privés et publics
22) Une pré-condition sur un opérateur définit les conditions préalables à l'exécution d'un tel opérateur

Exercice 1: (TAD)

On veut rédiger une spécification algébrique pour les nombre complexe. On définit les opérateurs suivants: un constructeur ***cpx*** qui prend deux nombres réels est crée un nombre complexe. On ajoute les inspecteurs suivants: ***partie-rl*** et ***partie-img*** qui retourne respectivement la partie réelle et la partie imaginaire d'un nombre complexe. Et enfin, on définit deux autres constructeurs qui sont ***add*** et ***sous*** permettant l'addition et la soustraction entre nombres complexes. Le TAD complexe utilise la sorte ***Réel*** et aussi les opérations **+** et **-** déjà prédéfinies dans ***Réel***. Proposez une spécification algébrique complète de ce TAD.

Genre: Complexe;
Sorte: Réel
Syntaxe:
cpx: Réel x Réel--> Complexe
partie-rl: Complexe-->Réel
partie-img: Complexe --> Réel
add: Complexe x Complexe --> Complexe
sous: Complexe x Complexe --> Complexe

2 points

Sémantique:
partie-rl(cpx(x,y))==x
partie-img(cpx(x,y))==y
partie-rl(add(C1,C2))==partie-rl(C1)+partie-rl(C2)
partie-img(add(C1,C2))==partie-img(C1)+partie-img(C2)
partie-rl(sous (C1,C2))==partie-rl(C1)-partie-rl(C2)
partie-img(sous(C1,C2))==partie-img(C1)-partie-img(C2)

3 points

Exercice 2: (POO)

Proposer une classe Complexe implémentant la spécification algébrique précédente en langage C++.

```
class Complexe{
      float re, im;
      public:
      Complexe(float re, float im);
      void Add(Complexe z1, Complexe z2);
      void Sous(Complexe z1, Complexe z2);
      float Partie_rl();
      float Partie_im();
};
```

2 points

```
//Définition
Complexe(float re, float im){
cout<<"Je suis créé"<<endl;
this.re=re;
this.im=im;
}

void Complexe::Add(Complexe z1, Complexe z2){
this->re=z1.re+z2.re;
this->im=z1.im+z2.im;
}
void Complexe::Sous(Complexe z1, Complexe z2){
this->re=z1.re-z2.re;
this->im=z1.im-z2.im;
}

float Complexe::Partie_rl(){
      return this->pr;
}
float Complexe::Partie_im(){
      return this->im;
}
```

3 points