Merci de respecter les cadres pour les réponses. Tout débordement ne sera pas pris en compte et pourra être pénalisé de 1 point maximum par exercice.

Question de cours (4 points)

1) Quelle est la différence entre une classe et un objet ? (0,5 pt)

2) Qu'est-ce que le principe d'encapsulation ? (0,5 pt)

3) Définir 2 propriétés d'une *Map* ? (1 pt)

4) Quels sont les avantages de la composition ? (1 pt)

5) Donner 2 relations possibles entre des classes. Expliquer en une phrase ces relations. (1 pt)

Exercice 1 : Etude d'une université (4 points)

Une université est composée de divers départements (sciences, technos, art…) regroupant des professeurs. On y trouve aussi des salles de classes comme par exemple la « UV-212 » ou encore « UV-424 ». Les salles peuvent être plus ou moins grandes et avoir 1 ou 2 portes. Je connais une amie qui a étudié à l'université Paris-Dauphine et y a rencontré M. Dupont, un professeur de maths.

Identifier sous forme de tableau : les classes, objets et attributs

Classes	*Objets*	*Attributs*

Exercice 2 (5 points)

Soit la classe suivante :

```
class Data
{
private:
    std::list<int> m_list;

public:
    Data(int _size); // remplir aléatoirement la liste
    ~Data(); // destructeur
    float moyenne(); // calcul la moyenne de la liste
    void afficher(); // affiche le contenu de la liste
};
```

1) Implémenter en C++ le constructeur qui remplit aléatoirement la liste avec des entiers

2) Implémenter en C++ la méthode *moyenne*()

3) Implémenter en C++ la méthode *afficher*()

4) Implémenter en C++ le destructeur

5) Ecrire les lignes du *main* qui créé un objet de type *Data* puis appelle la méthode *moyenne* et enfin la méthode *afficher*. Merci de libérer si nécessaire la mémoire allouée…

Exercice 3 : (2 points)

```
class Test
{
public:
    Test() { std::cout << "default ctor" << std::endl; }
    Test(int a) { std::cout << "overload ctor" << std::endl; }
    Test(int a, float b) { std::cout << "overload2 ctor" << std::endl; }
    ~Test() { std::cout << "default dtor" << std::endl;}
    int foo() { std::cout << "inside foo" << std::endl; }
};

int main()
{
    Test* t = new Test(0);
    Test t2;
    t->foo();
    return 0;
}
```

Ecrivez la sortie du programme (ce qui sera affiché à l'écran lors de son exécution).

Exercice 4 (5 points)

Ecrivez une classe *Compteur* qui comporte les méthodes suivantes :

1. un constructeur par défaut
2. un constructeur surchargé
3. une méthode d'incrémentation du compteur
4. une méthode de décrémentation du compteur
5. un accesseur pour récupérer la valeur du compteur

On implémentera les méthodes directement dans la classe pour plus de simplicité.

```
class Compteur {









};
```

Bon courage !

Annexe des fonctions pour le conteneur *liste* contenant des entiers :

void push_front(int valeur) ; // ajoute un élément en tête de liste
void push_back(int valeur) ; // ajoute un élément en fin de liste
void pop_front() ; // supprime le premier élément de la liste
void pop_back() ; // supprime le dernier élément de la liste
void insert(iterator position, int valeur) ; // ajoute un élement à une certaine position (itérateur)
iterator erase(iterator position) ; // supprime un élement à une certaine position et retourne un itérateur sur l'élément suivant
void clear() ; // supprime tous les éléments de la liste
int front() ; // retourne le premier élément de la liste
int back() ; // retourne le dernier élément de la liste