Examen de C++ Institut Galilée 2015

Le polycopié ainsi qu'une copie double de notes manuscrites ou tapées sont autorisées pendant l'examen. Pour l'exercice 1, seuls les fichiers `.hpp` et `.cpp` sont à rendre.

Exercice 1 *On cherche à créer une liste chaînée de véhicules. Les véhicules sont caractérisés par leur puissance, la vitesse et la couleur. En outre les véhicules peuvent être soit des voitures caractérisées par leur nombre de places, des car identifiées par le nombre de passagers ou des camions identifiées par le poids maximum. Les questions 2 et 3 ne sont pas absolument nécessaires pour continuer l'examen.*

1. *Créez des classes Vehicule, Voiture, Car et Camion avec des constructeurs sans arguments, des méthodes* `lire` *et* `ecrire` *sans arguments permettant de saisir au clavier et d'afficher à l'écran les caractéristiques de ces véhicules. Ici les classes héritent directement de Voiture, Car et Camion. Pour simplifier on suppose que l'utilisateur connaît les champs à rentrer. On suppose aussi que le programme (et non l'utilisateur détermine si le premier véhicule est un voiture, un car ou un camion). La classe Vehicule est ainsi définie*

```
class Vehicule
{
private:
  int puissance; int vitesse; har couleur;
public:
  Vehicule(void){}
  void lire(void);
  void ecrire();
};
```

Ce programme doit être réparti dans les fichiers suivants `listevehicules1.cpp` *contenant le main,* `vehicules1.hpp` *et* `vehicules1.cpp` *contenant les différentes classes.*

2. *Ajoutez deux nouvelles méthodes* `lire` *et* `ecrire` *à chacune de ces classes de façon à pouvoir faire la saisie et l'affichage au moyen des instructions suivantes, en faisant en sorte que la méthode* `lire` *n'utilise pas* `cin` *et que* `ecrire` *n'utilise pas* `cout`.

```
Voiture voiture1; voiture1.lire(cin); voiture1.ecrire(cout);
Car car1; car1.lire(cin); car1.ecrire(cout);
Camion camion1; camion1.lire(cin); camion1.ecrire(cout);
```

Ce programme doit être réparti dans les fichiers suivants `listevehicules2.cpp` *contenant le main,* `vehicules2.hpp` *et* `vehicules2.cpp` *contenant les différentes classes.*

3. *On souhaite maintenant faire en sorte que ce soit l'utilisateur qui détermine si le véhicule est une voiture, un car ou un camion. Plus précisément, on souhaite que l'appel de la méthode* `lire` *de Vehicule permettent à l'utilisateur d'entrer un champ voiture/car/camion et les caractéristiques supplémentaires liées à ce champ. Pour cela on introduit une classe intermédiaire entre Vehicule et Voiture,Car,Camion.*

```
class Lesvehicules
{
public:
  virtual void lire(istream & fluxIn)=0;
  virtual void lire(void)=0;
  virtual void ecrire(ostream & fluxOut)=0;
  virtual void ecrire(void)=0;
};
```

Cette classe ne dérive pas de Vehicule, mais on rajoute à la classe Vehicule un pointeur sur la classe Lesvehicules. Cette classe doit donc être définie avant la classe Vehicule. Il est nécessaire de penser à désallouer la place mémoire allouée. Ce programme doit être réparti dans les fichiers suivants `listevehicules3.cpp` *contenant le main,* `vehicules3.hpp` *et* `vehicules3.cpp` *contenant les différentes classes.*

4. *Créez une classe Objet contenant les méthodes virtuelles pures lire et ecrire (i.e. avec la syntaxe* `virtual ... = 0`*). Modifiez la classe Vehicule de façon à ce qu'elle hérite de la classe Objet, de façon que le programme continu à fonctionner. Si vous n'avez pas fait une des deux étapes précédentes, faites en sorte qu'il y ait une méthode lire et une méthode écrire qui puisse permettre de créer un objet éventuellement en imposant que l'objet créé soit nécessairement une voiture avec certaines caractéristiques prédéfinies. Ce programme doit être réparti dans les fichiers suivants* `listevehicules3.cpp` *contenant le main,* `vehicules4.hpp` *et* `vehicules4.cpp` *contenant les différentes classes relatives aux véhicules et* `objet4.hpp` *contenant la classe Objet.*

5. *Rajoutez la classe LienObjet en complétant le code suivant.*

```
class LienObjet
{
private:
  Objet * pobjet; LienObjet * pnext;
public:
  LienObjet(Objet * pobjet):pobjet(pobjet),pnext(NULL){}
  ~LienObjet() {}
  LienObjet * getNext() {...}
  void setNext(LienObjet * ptr) {...}
  Objet * getObjet() {...}
};
```

   Trouvez un moyen de créer dans le main une liste chaînée de deux éléments au moyen de cette classe LienObjet, puis de l'afficher. Ce programme doit être réparti dans les fichiers suivants `listevehicules5.cpp` *contenant le main,* `vehicules5.hpp` *et* `vehicules5.cpp` *contenant les différentes classes relatives aux véhicules et* `objet5.hpp` *contenant la classe Objet et* `listeobj5.hpp` *et éventuellement* `listeobj5.cpp` *pour la classe LienObjet.*

6. *Rajoutez la classe ListeObjet dont la déclaration est la suivante et insérez successivement 3 véhicules, avec insertion par le début de la liste et ensuite affichez l'ensemble de la liste (qui doit donc s'afficher dans l'ordre inverse. Dans cette étape on ne s'occupe pas de désallouer l'espace mémoire alloué.*

```
class ListeObjet
{
private:
  LienObjet *pdeb;
public:
  ListeObjet():pdeb(NULL){}
  void insererDebut(Objet * pobj);
};
```

   Ce programme doit être réparti dans les fichiers suivants `listevehicules6.cpp` *contenant le main,* `vehicules6.hpp` *et* `vehicules6.cpp` *contenant les différentes classes relatives aux véhicules et* `objet6.hpp` *contenant la classe Objet et* `listeobj6.hpp`*,* `listeobj6.cpp` *pour la classe LienObjet.*

7. *Modifiez la classe ListeObjet pour traiter le problème de la désallocation mémoire. Ce programme doit être réparti dans les fichiers suivants* `listevehicules7.cpp` *contenant le main,* `vehicules7.hpp` *et* `vehicules7.cpp` *contenant les différentes classes relatives aux véhicules et* `objet7.hpp` *contenant la classe Objet et* `listeobj7.hpp`*,* `listeobj7.cpp` *pour la classe LienObjet.*

8. *Rajouter un pointeur de LienObjet* `pfin` *et une méthode* `insererFin` *dans ListeObjet de façon à insérer les nouveaux en fin de liste, de telle sorte que l'affichage ne soit plus inversé. Modifiez en conséquence l'ensemble de la classe ListeObjet. Insérez successivement 3 véhicules en début de liste puis en fin de liste et affichez le résultat. Ce programme doit être réparti dans les fichiers suivants* `listevehicules8.cpp` *contenant le main,* `vehicules8.hpp` *et* `vehicules8.cpp` *contenant les différentes classes relatives aux véhicules et* `objet8.hpp` *contenant la classe Objet et* `listeobj8.hpp`*,* `listeobj8.cpp` *pour la classe LienObjet.*

Exercice 2

1. *Expliquez quelle pourrait être la raison en terme d'évolution future du programme, de définir des méthodes permettant de passer en référence un flux dans les méthodes lire et ecrire, plutôt que de réaliser des méthodes sans arguments ?*

2. *Expliquez aussi pourquoi une simple dérivation de la classe Voiture/Car/Camion vers la classe Vehicule n'aurait pas permis de répondre à la question 4 ?*

Exercice 3 *On cherche à simuler le fonctionnement d'une librairie. On dispose d'un fichier de livres (auteur, titre, code de localisation), un fichier de lecteurs (nom, mail, et pour chaque livre empreinté, la date de l'empreint et un index). Cet index permet de retrouver le livre dans le fichier de livres. On souhaite envisager le cas où il y a plusieurs exemplaires du même livre. On souhaite d'une part pour chaque livre identifié par son titre, savoir quand est-ce que chaque exemplaire sera rendu et d'autre part pour chaque lecteur lui envoyer, le cas échéant, un mail 3 jours avant la date limite pour lui demander de rendre les livres qu'il doit rendre. Proposez des classes et des méthodes permettant de résoudre ce problème.*