Commencé le	jeudi 2 novembre 2023, 21:39
État	Terminé
Terminé le	jeudi 2 novembre 2023, 21:39
Temps mis	13 s
Note	**0,00** sur 100,00

Question 1

Non répondue

Non noté

Si nécessaire, inscrivez vos suppositions ici, en précisant pour chaque supposition le numéro de la question concernée.

Question 2

Non répondue

Noté sur 2,00

Si une classe ne possede aucun constructeur, il est impossible d'instancier un objet de cette classe.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 3

Non répondue

Noté sur 2,00

```
class Vehicule{
   public:
     Vehicule ();
     Vehicule(int = 0);
};
Vehicule bus;
```

L'objet bus peut-il être instancié ?

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 4

Non répondue

Noté sur 2,00

Identifiez l'implémentation des constructeurs de ces classes.

```
class Moteur{
  public:
   Moteur (double puissance);
  private:
   double puissance_;
 };
class Vehicule{
  public:
    Vehicule (int id, double puissance);
  private:
   Moteur   moteur_;
   int id_;
};
class Autobus: public Vehicule{
  public:
   Autobus (int id, double puissance, string proprio);
  private:
   string proprietaire_;
};
```

Veuillez choisir au moins une réponse.

- A. Moteur(double puissance): puissance_(puissance){}
- B. Autobus (int id, double puissance, string proprio):Vehicule (id, puissance){ proprietaire_= proprio; }
- C. Vehicule::Vehicule (int id = 0 , double puissance = 0.0):id_(id), moteur_(puissance){}
- D. Vehicule::Vehicule (int id, double puissance):id_(id), moteur_(puissance){}
- E. Moteur:: Moteur(double puissance = 0): puissance_(puissance){}
- F. Vehicule (int id, double puissance):id_(id), moteur_(puissance){}
- G. Autobus::Autobus (int id, double puissance, string proprio): Vehicule (id, puissance), proprietaire_(proprio){}
- H. Autobus::Autobus (int id, double puissance, string proprio){ Vehicule (id, puissance); proprietaire_= proprio; }
- I. Moteur:: Moteur(double puissance): puissance_(puissance){}

Votre réponse est incorrecte.

Les réponses correctes sont : Moteur:: Moteur(double puissance): puissance_(puissance){}, Vehicule::Vehicule (int id, double puissance):id_(id), moteur_(puissance){}, Autobus::Autobus (int id, double puissance, string proprio): Vehicule (id, puissance), proprietaire_(proprio){}

Question 5

Non répondue

Noté sur 4,00

Détermimer pour chaque classe **CompteCellulaire**, s'il s'agit d'une composition, une agrégation par référence ou une agrégation par pointeur de la classe **Appareil**.

```
class Appareil
{};
class CompteCellulaire{
public:
    CompteCellulaire( const Appareil & app): app_(app){}
private:
    Appareil app_;
};

class CompteCellulaire2{
public:
    CompteCellulaire2( const Appareil & app): app_(make_unique<Appareil>(app)){}
private:
    unique_ptr<Appareil> app_;
};

class CompteCellulaire3{
public:
 CompteCellulaire3( Appareil & app): app_(app){}
 private:
  Appareil & app_;
};
class CompteCellulaire4{
public:
  CompteCellulaire4(const shared_ptr<Appareil> & app): app_(app){}
private:
  shared_ptr<Appareil>  app_;
};
```

Question 6

Non répondue

Noté sur 2,00

L'attribut de la classe **Semaine** est s**hared_ptr <Cours> cours_** et la classe **Semaine** est une agrégation de la classe **Cours**.

Identifier les définitions et implémentations possibles du constructeur de la classe **Semaine**.

Veuillez choisir au moins une réponse.

a. `Semaine(shared_ptr<Cours> cours) { *cours_ = *cours; }`

b. `Semaine(Cours cours) { cours_= &cours; }`

c. `Semaine(shared_ptr<Cours> cours) { cours_= make_shared <Cours>(); *cours_ = *cours; }`

d. `Semaine(shared_ptr<Cours> cours) : cours_(cours) {}`

e. `Semaine(shared_ptr<Cours> cours) { cours_= &cours; }`

f. `Semaine(shared_ptr<Cours> cours) { cours_ = make_shared<Cours>(*cours); }`

g. `Semaine(shared_ptr<Cours> cours) { cours_= cours; }`

Votre réponse est incorrecte.

Les réponses correctes sont :

```
Semaine(shared_ptr<Cours> cours) { cours_= cours; }
```

,

```
Semaine(shared_ptr<Cours> cours) : cours_(cours) {}
```

Question 7

Non répondue

Noté sur 2,00

Identifiez les déclarations suivantes.

unique_ptr< unique_ptr<Vehicule []> [] > liste;	Choisir…
unique_ptr< unique_ptr<Vehicule> [] > liste;	Choisir…
unique_ptr< unique_ptr<unique_ptr<Vehicule> []> [] > liste;	Choisir…
unique_ptr<Vehicule> liste;	Choisir…

Votre réponse est incorrecte.

La réponse correcte est : unique_ptr< unique_ptr<Vehicule []> [] > liste; → tableau 2D dynamique d'objets Vehicule, unique_ptr< unique_ptr<Vehicule> [] > liste; → tableau dynamique de pointeurs à des objets Vehicule, unique_ptr< unique_ptr<unique_ptr<Vehicule> []> [] > liste; → tableau 2D dynamique de pointeurs à des objets Vehicule, unique_ptr<Vehicule> liste; → pointeur à un objet Vehicule

Question 8

Non répondue

Noté sur 2,00

En lisant la définition d'une classe, vous trouvez l'attribut **b_** suivant:

```
class A
{
//...
private:
   unique_ptr<B> b_;
};
```

Quel(s) énoncé(s) sont possibles par rapport à l'allocation du pointeur **b_**?

Veuillez choisir au moins une réponse.

- a. Il correspond à un tableau de pointeurs vers des objets de type **B** réservé dans la pile.
- b. Il correspond à un tableau d'objets de type **B** dans le tas.
- c. Il pointe vers un seul objet de type **B** réservé dans le tas.
- d. Il correspond à un tableau de pointeurs réservé dans le tas où chaque élément pointe vers un espace mémoire de type **B** dans le tas.

Votre réponse est incorrecte.

La réponse correcte est : Il pointe vers un seul objet de type **B** réservé dans le tas.

Question 9

Non répondue

Noté sur 4,00

Soit les classes suivantes

```
class Ingredient {
public:
  Ingredient(string nom, int nbCalories);
  int getNbCalories() const;

private:
  string nom_;
  int nbCalories_;
};

Ingredient::Ingredient(string nom, int nbCalories)
  : nom_(nom), nbCalories_(nbCalories) {}

int Ingredient::getNbCalories() const {
  return nbCalories_;
}
class Repas {
  public:

  private:
  vector<shared_ptr<Ingredient>> ingredients_;
};
```

Donnez les signatures des méthodes suivantes de la classe Repas.

[ ] operator == ([ ]);

[ ] operator += ([ ]);

Donnez la signature la fonction globale suivantem pour la classe Repas.

[ ] [ ] operator << ([ ] , [ ]);

Votre réponse est incorrecte.

La réponse correcte est : Soit les classes suivantes

```
class Ingredient {
public:
  Ingredient(string nom, int nbCalories);
  int getNbCalories() const;

private:
  string nom_;
  int nbCalories_;
};

Ingredient::Ingredient(string nom, int nbCalories)
  : nom_(nom), nbCalories_(nbCalories) {}

int Ingredient::getNbCalories() const {
  return nbCalories_;
}
class Repas {
  public:

  private:
  vector<shared_ptr<Ingredient>> ingredients_;
};
```

Donnez les signatures des méthodes suivantes de la classe Repas.

[bool]operator == ([const Repas & r]);

[Repas &] operator += ([const shared_ptr<Ingredient> & i]);

Donnez la signature la fonction globale suivantem pour la classe Repas.

[friend] [ostream &] operator << ([ostream & out] , [const Repas & repas]);

Question 10

Non répondue

Noté sur 5,00

Soit la classe Personne, pour chaque instruction suivante, identifiez la signature des méthodes ou des fonctions globales si on a les déclarations suivantes:

Personne p1, p2, p3;

p1 *=10;	Choisir…
cout << (p1==p2);	Choisir…
cout << p1;	Choisir…
p1 = 100 + p2;	Choisir…
cout << (p1 != p2);	Choisir…
p1 = p1+ p2;	Choisir…
p1 += p2 += p3;	Choisir…
cout << (40 == p1);	Choisir…
cout << (p1 <= 100);	Choisir…
p1 = p2 + 40;	Choisir…

Votre réponse est incorrecte.

La réponse correcte est :

p1 *=10; → Personne & operator *= (float);,

cout << (p1==p2); → bool operator == (const Personne &);,

cout << p1; → friend ostream & operator < < (ostream &, const Personne &_);,

p1 = 100 + p2; → friend Personne operator + (float,const Personne &);,

cout << (p1 != p2); → bool operator != (const Personne &);,

p1 = p1+ p2; → Personne operator + (const Personne &);,

p1 += p2 += p3; → Personne & operator += (const Personne &);,

cout << (40 == p1); → friend bool operator == (float, const Personne &);,

cout << (p1 <= 100); → bool operator < = (float);,

p1 = p2 + 40; → Personne operator + (float);

Question 11

Non répondue

Noté sur 1,00

Quel est le meilleur type pour déclarer un objet afin de sauver de l'espace mémoire et du temps d'exécution ?

Veuillez choisir une réponse.

- A. vector <vehicule>
- B. vector <shared_ptr<vehicule>>

Votre réponse est incorrecte.

La réponse correcte est : vector <shared_ptr<vehicule>>

Question 12

Non répondue

Noté sur 1,00

Combien de constructeurs la classe Vector possède -t-elle dans la STL?

Réponse : ✖

La réponse correcte est : 10

Question 13

Non répondue

Noté sur 1,00

L'opérateur d'affectation de la classe Vector fait un deep copie des éléments du conteneur.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Vrai ».

Question 14

Non répondue

Noté sur 1,00

La classe Vector possède deux attributs: le nombre d'éléments et le pointeur du tableau dynamique.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 15

Non répondue

Noté sur 1,00

Le paramètre de la méthode push_back du conteneur vector est transmis par référence.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 16

Non répondue

Noté sur 4,00

Soit les classes suivantes

```
class Personne{
  public:
     virtual  void travailler() = 0;
     virtual  void reposer() = 0;
};
class Employe{
  public:
    virtual void faireTempsSupplementaire() = 0;
 private:
     string nom_;
 };

class Salarie: public Personne, public Employe {
  public:
    void travailler() override;
    void faireTempsSupplementaire() override;
 };
class Directeur : public Salarie{
 public:
    void reposer() override;
 private:
    int matricule_;
};
```

Identifiez le genre de classe.

Personne

Employe

Salarie

Directeur

Votre réponse est incorrecte.

La réponse correcte est : Soit les classes suivantes

```
class Personne{
  public:
     virtual  void travailler() = 0;
     virtual  void reposer() = 0;
};
class Employe{
  public:
    virtual void faireTempsSupplementaire() = 0;
 private:
     string nom_;
 };

class Salarie: public Personne, public Employe {
  public:
    void travailler() override;
    void faireTempsSupplementaire() override;
 };
class Directeur : public Salarie{
 public:
    void reposer() override;
 private:
    int matricule_;
};
```

Identifiez le genre de classe.

Personne [Interface]

Employe [classe abstraite]

Salarie [classe abstraite]

Directeur [classe concrète]

Question 17

Non répondue

Noté sur 4,00

Soit le code C++ suivant :

```
class AA {
public:
    AA() { cout << "AA() "; }
private:
    int attC_;
}
class CC {
public:
    CC() { cout << "CC() "; }
private:
    int attE_;
};
class BB {
public:
    BB() {
        cout << "BB() ";
        pA_ =  make_unique< AA>();
    }
private:
    CC  attB_;
    unique_ptr<AA> pA_;
};
class DD : public BB {
public:
    DD(CC & attribut):attD_(attribut) { cout << "DD() "; }
private:
    CC & attD_;

};
```

Quel est l'affichage de la ligne suivante ?

unique_ptr<BB> p1 = make_unique<BB>();

Quel est l'affichage de la ligne suivante ?

CC objetCC;

unique_ptr<DD> p2 = make_unique<DD>(objetCC);

Quelle est la relation entre les classes ?

AA et CC

BB et AA

BB et CC

DD et BB

DD et CC

Votre réponse est incorrecte.

La réponse correcte est :
Soit le code C++ suivant :

```
class AA {
public:
    AA() { cout << "AA() "; }
private:
    int attC_;
}
class CC {
public:
    CC() { cout << "CC() "; }
private:
    int attE_;
};
class BB {
public:
    BB() {
        cout << "BB() ";
        pA_ =  make_unique< AA>();
    }
private:
    CC  attB_;
    unique_ptr<AA> pA_;
};
class DD : public BB {
public:
    DD(CC & attribut):attD_(attribut) { cout << "DD() "; }
private:
    CC & attD_;

};
```

Quel est l'affichage de la ligne suivante ?

unique_ptr<BB> p1 = make_unique<BB>();

[CC()] [BB()] [AA()]

Quel est l'affichage de la ligne suivante ?

CC objetCC;
unique_ptr<DD> p2 = make_unique<DD>(objetCC);
[CC()] [CC()] [BB()] [AA()] [DD()]

Quelle est la relation entre les classes ?
AA et CC [pas de relation]
BB et AA [Composition]
BB et CC [Composition]
DD et BB [Héritage]
DD et CC [Aggrégation]

Question 18

Non répondue

Noté sur 2,00

Soit le code suivant:

```
#include <iostream>
using namespace std;

class MatelasBase {
public:
    MatelasBase() {}

    double getPrix() const {  return 1000;  }
};

class MatelasSophistique : public MatelasBase {
public:
    MatelasSophistique(double facteur): facteur_(facteur) {}

    double getPrix() const {
       // votre choix
    }
private:
    double facteur_;
};

int main() {
    MatelasSophistique ms(1.2);
    cout << ms.getPrix() << "$"; // Devrait afficher 1200$;
}
```

Identifiez la méthode getPrix() de **MatelasSophistique** qui applique le facteur sur le prix indiqué par la classe de base.

Veuillez choisir une réponse.

- A. return getPrix() *facteur_;
- B. return MatelasSophistique ::getPrix() *facteur_;
- C. return 1000 *facteur_;
- D. this->getPrix();
- E. return MatelasBase::getPrix() *facteur_;

Votre réponse est incorrecte.

La réponse correcte est :
return MatelasBase::getPrix() *facteur_;

Question 19

Non répondue

Noté sur 2,00

Soit les classes suivantes

```
class AA{
public:
    void f1(){};
    void f2(){};
    void f3(){};
    void f4(){};
    void f5(){};
};
class BB: public AA{
public:
    void f2(){};
    void f3(){};
    void f4(){};
    void f5(){};
};
class CC: public AA{
public:
    void f2(){};
    void f3(){};
    void f4(){};
};

 class DD: public BB, public CC{
    public:
      void f3(){};
 };
```

Dites quelle affirmation est vraie lorsqu'on exécute le code suivant:

DD d;

d.f1();

Veuillez choisir une réponse.

- A. Ce code produit une erreur de compilation liée au problème du diamant.
- B. Ce code ne produit aucune erreur de compilation.
- C. Ce code produit une erreur de compilation liée à l'ambiguïté du nom.

Votre réponse est incorrecte.

La réponse correcte est : Ce code produit une erreur de compilation liée au problème du diamant.

Question 20

Non répondue

Noté sur 3,00

Soit les classes suivantes

```
class Vehicule{
};
class Autobus: public Vehicule{
};
```

Donnez le comportement des instructions du code suivant:

```
Autobus bus;
Vehicule  vehicule = bus;

bus = static_cast<Autobus> (vehicule);

unique_ptr<Autobus> busptr = make_unique<Autobus> ();
Vehicule * vehiculeptr;
vehiculeptr = static_cast<Vehicule *> (busptr.get());

Autobus * busptr2 = static_cast<Autobus *> ( vehiculeptr);

vehiculeptr = &bus;
busptr2 = static_cast<Autobus *> (vehiculeptr);

busptr2 = static_cast<Autobus *> (&vehicule);
```

Votre réponse est incorrecte.

La réponse correcte est : Soit les classes suivantes

```
class Vehicule{
};
class Autobus: public Vehicule{
};
```

Donnez le comportement des instructions du code suivant:

```
Autobus bus;
Vehicule  vehicule = bus;

bus = static_cast<Autobus> (vehicule);[Erreur de compilation]

unique_ptr<Autobus> busptr = make_unique<Autobus> ();[Exécution normale]
Vehicule * vehiculeptr;
vehiculeptr = static_cast<Vehicule *> (busptr.get()); [Exécution normale]

Autobus * busptr2 = static_cast<Autobus *> ( vehiculeptr);[Exécution indéfini]

vehiculeptr = &bus;[Exécution normale]
busptr2 = static_cast<Autobus *> (vehiculeptr);[Exécution normale]

busptr2 = static_cast<Autobus *> (&vehicule); [Exécution indéfini]
```

Question 21

Non répondue

Noté sur 2,00

```
class Vehicule{
};
class Autobus: public Vehicule{
  public:
     int  getId();
};
int main()
{
  vector<unique_ptr< Vehicule > > vehicules;
  vehicules.push_back(make_unique<Autobus>());
```

Quelle est l'instruction pour faire appel à la méthode getId() du permier élément du conteneur vehicules ?

Veuillez choisir une réponse.

- A. dynamic_cast <Autobus*> (vehicules[0].get())->getId()
- B. static_cast <Autobus> (vehicules[0])->getId()
- C. dynamic_cast <Vehicule> (*vehicules[0]).getId()
- D. static_cast <Autobus*> (vehicules[0].get())->getId()
- E. vehicules[0]->getId()
- F. dynamic_cast <Vehicule*> (vehicules[0].get())->getId()

Votre réponse est incorrecte.

La réponse correcte est : static_cast <Autobus*> (vehicules[0].get())->getId()

Question 22

Non répondue

Note de 0,00 sur 5,00

Soit la classe ArretAutobus

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
```

Écrire la méthode operator == qui compare le nom des rues.

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 23

Non répondue

Note de 0,00 sur 12,00

Soit les classes ArretAutobus et LigneAutobus.

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
class LigneAutobus {
public:
    LigneAutobus(int numero) : numero_(numero) {}
    bool ajouterArret(shared_ptr<ArretAutobus> a);
    unsigned getNbArrets(){ return arrets_.size(); }
    unsigned getNumero(){ return numero_;}
private:
// agrégation de pointeurs à des objets ArretAutobus
    vector<shared_ptr<ArretAutobus> > arrets_;
    unsigned numero_;
};
```

Écrire la méthode ajouterArret qui ajoute des arrêts dans le vector pointeurs à des objets ArretsAutobus. Il faut évidement ajouter un arrêt que s'il n'existe pas dans le vector.

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 24

Non répondue

Note de 0,00 sur 7,00

Écrire la définition et l'implémentation de l'interface Affichable qui permet d'afficher toutes ses classes dérivées en utilisant l'opérateur de flux de sortie. Voici un exemple de hiérarchie de classe dérivant de Affichable qui a déjà été implémentée dans le code:

```
class Autobus : public Affichable{
public:
    Autobus(char id);
     ~Autobus() override;
    void setLigne(shared_ptr<LigneAutobus> ligne) { ligne_ = ligne; }
    char getId() { return id_; }
    shared_ptr<LigneAutobus> getLigne()const;
    ostream& afficher(ostream& out) const  override;
private:
    shared_ptr<LigneAutobus> ligne_; // agrégation d'une ligne d'autobus
    char id_;
};
class AutobusEssence: public Autobus{
    public:
        AutobusEssence( char id, float prix);
        ~AutobusEssence() override;
        float getCoutEssence()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixEssence_;
};
class AutobusElectrique: public Autobus{

    public:
        AutobusElectrique( char id, float prix);
         ~AutobusElectrique() override;
        float getCoutElectrique()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixKwh_;
};
```

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 25

Non répondue

Note de 0,00 sur 5,00

Soit les classes suivantes

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
class LigneAutobus {
public:
    LigneAutobus(int numero) : numero_(numero) {}
    bool ajouterArret(shared_ptr<ArretAutobus> a);
    unsigned getNbArrets(){ return arrets_.size(); }
    unsigned getNumero(){ return numero_;}
private:
// agrégation de pointeurs à des objets ArretAutobus
    vector<shared_ptr<ArretAutobus> > arrets_;
    unsigned numero_;
};
class Autobus : public Affichable{
public:
    Autobus(char id);
     ~Autobus() override;
    void setLigne(shared_ptr<LigneAutobus> ligne) { ligne_ = ligne; }
    char getId() { return id_; }
    shared_ptr<LigneAutobus> getLigne()const;
    ostream& afficher(ostream& out) const  override;
private:
    shared_ptr<LigneAutobus> ligne_; // agrégation d'une ligne d'autobus
    char id_;
};
class AutobusEssence: public Autobus{
    public:
        AutobusEssence( char id, float prix);
        ~AutobusEssence() override;
        float getCoutEssence()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixEssence_;
};
class AutobusElectrique: public Autobus{

    public:
        AutobusElectrique( char id, float prix);
         ~AutobusElectrique() override;
        float getCoutElectrique()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixKwh_;
};
class ReseauAutobus  {
public:
    ReseauAutobus(const string & compagnie = "");
    void ajouterAutobus(unique_ptr<Autobus> & autobus);
    bool setLigneAutobus(char id, shared_ptr<LigneAutobus> ligne);
    float  coutFlotteAutobus();
    friend ostream& operator<<(ostream& out, const ReseauAutobus & a);
private:
    vector<unique_ptr<Autobus>> autobus_; // composition d'autobus
    string compagnie_;
};
```

Écrire le constructeur de la classe ReseauAutobus.

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 26

Non répondue

Note de 0,00 sur 5,00

Soit les classes suivantes:

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
class LigneAutobus {
public:
    LigneAutobus(int numero) : numero_(numero) {}
    bool ajouterArret(shared_ptr<ArretAutobus> a);
    unsigned getNbArrets(){ return arrets_.size(); }
    unsigned getNumero(){ return numero_;}
private:
// agrégation de pointeurs à des objets ArretAutobus
    vector<shared_ptr<ArretAutobus> > arrets_;
    unsigned numero_;
};
class Autobus : public Affichable{
public:
    Autobus(char id);
     ~Autobus() override;
    void setLigne(shared_ptr<LigneAutobus> ligne) { ligne_ = ligne; }
    char getId() { return id_; }
    shared_ptr<LigneAutobus> getLigne()const;
    ostream& afficher(ostream& out) const  override;
private:
    shared_ptr<LigneAutobus> ligne_; // agrégation d'une ligne d'autobus
    char id_;
};
class AutobusEssence: public Autobus{
    public:
        AutobusEssence( char id, float prix);
        ~AutobusEssence() override;
        float getCoutEssence()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixEssence_;
};
class AutobusElectrique: public Autobus{

    public:
        AutobusElectrique( char id, float prix);
         ~AutobusElectrique() override;
        float getCoutElectrique()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixKwh_;
};
class ReseauAutobus  {
public:
    ReseauAutobus(const string & compagnie = "");
    void ajouterAutobus(unique_ptr<Autobus> & autobus);
    bool setLigneAutobus(char id, shared_ptr<LigneAutobus> ligne);
    float  coutFlotteAutobus();
    friend ostream& operator<<(ostream& out, const ReseauAutobus & a);
private:
    vector<unique_ptr<Autobus>> autobus_; // composition d'autobus
    string compagnie_;
};
```

Écrire la méthode ajouterAutobus de la classe ReseauAutobus qui ajoute un pointeur intelligent d'autobus dans le vector. Il est inutile de vérifier si l'autobus est dans le ReseauAutobus.

Réponse : (régime de pénalités : 0. %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 27

Non répondue

Note de 0,00 sur 7,00

Soit les classes suivantes:

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
class LigneAutobus {
public:
    LigneAutobus(int numero) : numero_(numero) {}
    bool ajouterArret(shared_ptr<ArretAutobus> a);
    unsigned getNbArrets(){ return arrets_.size(); }
    unsigned getNumero(){ return numero_;}
private:
// agrégation de pointeurs à des objets ArretAutobus
    vector<shared_ptr<ArretAutobus> > arrets_;
    unsigned numero_;
};
class Autobus : public Affichable{
public:
    Autobus(char id);
     ~Autobus() override;
    void setLigne(shared_ptr<LigneAutobus> ligne) { ligne_ = ligne; }
    char getId() { return id_; }
    shared_ptr<LigneAutobus> getLigne()const;
    ostream& afficher(ostream& out) const  override;
private:
    shared_ptr<LigneAutobus> ligne_; // agrégation d'une ligne d'autobus
    char id_;
};
class AutobusEssence: public Autobus{
    public:
        AutobusEssence( char id, float prix);
        ~AutobusEssence() override;
        float getCoutEssence()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixEssence_;
};
class AutobusElectrique: public Autobus{

    public:
        AutobusElectrique( char id, float prix);
         ~AutobusElectrique() override;
        float getCoutElectrique()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixKwh_;
};
class ReseauAutobus  {
public:
    ReseauAutobus(const string & compagnie = "");
    void ajouterAutobus(unique_ptr<Autobus> & autobus);
    bool setLigneAutobus(char id, shared_ptr<LigneAutobus> ligne);
    float  coutFlotteAutobus();
    friend ostream& operator<<(ostream& out, const ReseauAutobus & a);
private:
    vector<unique_ptr<Autobus>> autobus_; // composition d'autobus
    string compagnie_;
};
```

Écrire la surcharge de l'opérateur << de la classe ReseauAutobus en faisant appel à l' opérateur << des autres classes.

Par exemple:

Test	Résultat

`ReseauAutobus stm("STM");`
`unique_ptr<Autobus> aut1 = make_unique<AutobusEssence>('Z', 1.55);`
`unique_ptr<Autobus> aut2 =make_unique<AutobusElectrique>('B', 2);`
`stm.ajouterAutobus (aut1);`
`stm.ajouterAutobus (aut2);`
`cout << stm;`
	`Compagnie STM`
	`ID Z`
	`Autobus Essence`
	`ID B`
	`Autobus Electrique`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**

Question 28

Non répondue

Note de 0,00 sur 12,00

Soit les classes suivantes

```
class ArretAutobus {
public:
    ArretAutobus(string rue ="") : rue_(rue) {}
    bool operator == (const ArretAutobus & a);
private:
    string rue_;
};
class LigneAutobus {
public:
    LigneAutobus(int numero) : numero_(numero) {}
    bool ajouterArret(shared_ptr<ArretAutobus> a);
    unsigned getNbArrets(){ return arrets_.size(); }
    unsigned getNumero(){ return numero_;}
private:
// agrégation de pointeurs à des objets ArretAutobus
    vector<shared_ptr<ArretAutobus> > arrets_;
    unsigned numero_;
};
class Autobus : public Affichable{
public:
    Autobus(char id);
     ~Autobus() override;
    void setLigne(shared_ptr<LigneAutobus> ligne) { ligne_ = ligne; }
    char getId() { return id_; }
    shared_ptr<LigneAutobus> getLigne()const;
    ostream& afficher(ostream& out) const  override;
private:
    shared_ptr<LigneAutobus> ligne_; // agrégation d'une ligne d'autobus
    char id_;
};
class AutobusEssence: public Autobus{
    public:
        AutobusEssence( char id, float prix);
        ~AutobusEssence() override;
        float getCoutEssence()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixEssence_;
};
class AutobusElectrique: public Autobus{

    public:
        AutobusElectrique( char id, float prix);
         ~AutobusElectrique() override;
        float getCoutElectrique()const;
        ostream& afficher(ostream& out) const  override;
    private:
       float  prixKwh_;
};
class ReseauAutobus  {
public:
    ReseauAutobus(const string & compagnie = "");
    void ajouterAutobus(unique_ptr<Autobus> & autobus);
    bool setLigneAutobus(char id, shared_ptr<LigneAutobus> ligne);
    float coutFlotteAutobus();
    friend ostream& operator<<(ostream& out, const ReseauAutobus & a);
private:
    vector<unique_ptr<Autobus>> autobus_; // composition d'autobus
    string compagnie_;
};
```

Écrire la méthode coutFlotteAutobus() de la classe ReseauAutobus qui

- parcourt l'attribut autobus_
- calcule le cout total des autobus Essence et autobus Electrique en faisant la somme des retours d'appel des méthodes getCoutEssence() et getCoutElectrique() pour chaque autobus.
- retourne le cout total.

Attention, il faut vérifier que l'autobus est associé à une ligne d'autobus pour faire le calcul.

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?

Falling back to raw text area.

▸ **Montrer / masquer la solution de l'auteur de la question (Cpp)**