Commencé le	vendredi 3 novembre 2023, 14:58
État	Terminé
Terminé le	vendredi 3 novembre 2023, 14:59
Temps mis	33 s
Note	**0,50** 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

Partiellement correct

Note de 0,50 sur 5,00

Soient les classes et les fonctions globales suivantes:

```
class AA{
public:
   AA();
   AA(int);
   AA(const AA &)
};
class BB: public AA {
public:
   BB ();
   BB(int);
   BB(const BB &);
   BB(const AA & );
 };
void fct1(BB objet);
void fct2(const BB & objet);
void fct3 (AA & objet);
```

Quel est le constructeur qui est appelé lors des déclarations et des instructions suivantes ?

```
AA objA1;  AA();   ✔
AA objA2(objA1);   ✖
BB objB3 (1);   ✖
BB objB4 = objB3;   ✖

fct1(1);   ✖
fct1(objA1);   ✖
fct1(objB3);   ✖
fct2 (objB3);   ✖
fct3(objB4);   ✖
fct3(objB3);   ✖
```

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 1.

La réponse correcte est :

Soient les classes et les fonctions globales suivantes:

```
class AA{
public:
   AA();
   AA(int);
   AA(const AA &)
};
class BB: public AA {
public:
   BB ();
   BB(int);
   BB(const BB &);
   BB(const AA & );
 };
void fct1(BB objet);
void fct2(const BB & objet);
void fct3 (AA & objet);
```

Quel est le constructeur qui est appelé lors des déclarations et des instructions suivantes ?

```
AA objA1; [AA();]
AA objA2(objA1);[AA(const AA &)]
BB objB3 (1);[BB(int);]
BB objB4 = objB3;[BB(const BB & );]

fct1(1); [BB(int);]
fct1(objA1);[BB(const AA & );]
fct1(objB3);[BB(const BB & );]
fct2 (objB3);[Aucun constructeur]
fct3(objB4); [Aucun constructeur]
fct3(objB3);[Aucun constructeur]
```

Question 3

Non répondue

Noté sur 3,00

Soient les signatures des fonctions suivantes:

```
void fonction(const Cours & c);// A
void fonction(Cours * c);// B
```

Et les déclarations suivantes

```
shared_ptr <Cours> c1 = make_shared<Cours> ();
Cours* c2;
Cours c3;
```

Quelle fonction est exécutée lors des appels suivants ?

fonction(c2); Choisir…

fonction(&c3); Choisir…

fonction (*c2); Choisir…

fonction(c1.get()); Choisir…

fonction(c3); Choisir…

fonction(c1); Choisir…

Votre réponse est incorrecte.

La réponse correcte est : fonction(c2); → B, fonction(&c3); → B, fonction (*c2); → A, fonction(c1.get()); → B, fonction(c3); → A, fonction(c1); → Ni A, ni B

Question 4

Non répondue

Noté sur 2,00

Soit la classe AA, quelle doit être la définition de l'opérateur += ?

Veuillez choisir une réponse.

1. `AA  operator += (const AA & );`
2. `AA & operator += (const AA & );`
3. `void operator+=  (const AA &,const AA & );`

Votre réponse est incorrecte.

La réponse correcte est :

```
AA & operator += (const AA & );
```

Question 5

Non répondue

Noté sur 2,00

Si une classe ne fait aucune allocation dynamique, quels sont les méthodes et les opérateurs que l'on peut utiliser même s'ils ne sont pas implémentés dans la classe ?

Attention une mauvaise réponse entraîne une pénalité.

Veuillez choisir au moins une réponse.

- [ ] 1. operator ==
- [ ] 2. operator =
- [ ] 3. operator +
- [ ] 4. operator -
- [ ] 5. operator <<
- [ ] 6. constructeur de copie

Votre réponse est incorrecte.

Les réponses correctes sont :
constructeur de copie,

operator =

Question 6

Non répondue

Noté sur 2,00

La classe BB est une agrégation de la classe AA.

```
class AA{
};
class BB{
  public:
      BB( AA & a):attribut_(a){};
  private:
      AA & attribut_;
};
```

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Vrai ».

Question 7

Non répondue

Noté sur 2,00

Les signatures des fonctions globales sont équivalentes

1. void uneFonction(const Employee& emp)
2. void uneFonction(Employee const & emp)

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 8

Non répondue

Noté sur 2,00

Si l'on exécute ce programme, il n'y a aucune erreur de compilation.

```
class AA{
  public:
     void methode();
 };
 class BB {
   public:
      void methode() const { attribut_.methode();}
    private:
      AA attribut_;
 };
int main() {
  BB unObjetB;
  unObjetB.methode();
  return 0;
}
```

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 9

Non répondue

Noté sur 2,00

La méthode clear() retire tous les éléments d'un vecteur et ramène sa capacité à 0.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 10

Non répondue

Noté sur 2,00

Soit la déclaration suivante:

```
unique_ptr <unique_ptr <unique_ptr <Cours>[]> [] > liste;
```

Quelles sont les instructions pour allouer dynamiquement de l'espace mémoire à la variable liste ?

Veuillez choisir au moins une réponse.

1.
```
liste = make_unique <unique_ptr <unique_ptr <Cours>[]> []> (10);
for (int i = 0; i < 10; i++)
    liste[i] = make_unique< unique_ptr<Cours> []> (5);
liste[0][0] = make_unique<Cours>();
```

2.
```
liste = make_unique <unique_ptr <unique_ptr <Cours> > > (10);
for (int i = 0; i < 10; i++)
    liste[i] = make_unique< unique_ptr<Cours>> (5);
liste[0][0] = make_unique<Cours>();
```

3.
```
liste = make_unique <unique_ptr <unique_ptr <Cours>[]> [10]> ;
for (int i = 0; i < 10; i++)
    liste[i] = make_unique< unique_ptr<Cours> [5]> ;
liste[0][0] = make_unique<Cours>();
```

4.
```
liste = make_unique <unique_ptr <unique_ptr <Cours>[]> []> (10);
for (int i = 0; i < 10; i++)
    liste[i] = make_unique< unique_ptr<Cours>> (5);
liste[0][0] = make_unique<Cours>();
```

Votre réponse est incorrecte.

La réponse correcte est :

```
liste = make_unique <unique_ptr <unique_ptr <Cours>[]> []> (10);
for (int i = 0; i < 10; i++)
    liste[i] = make_unique< unique_ptr<Cours> []> (5);
liste[0][0] = make_unique<Cours>();
```

Question 11

Non répondue

Noté sur 4,00

Compléter le code suivant:

```
class CostumeHalloween {
public:
        CostumeHalloween(const string& id) : id_(id) {}
        string getId() const { return id_; }
        void operator++() { nbFoisLoue_++; }

private:
        string id_;
        unsigned int nbFoisLoue_;
};
class MagasinCostumesHalloween {
public:
        const CostumeHalloween* louerCostume(const string& id) {
                for(                                        costume: marchandises_){
                        if (                getId() == id) {
                                ++                  ;
                                return                  ;
                        }
                }
                return nullptr;
        }

private:
        vector<unique_ptr<CostumeHalloween>> marchandises_;
};
```

Votre réponse est incorrecte.

La réponse correcte est :
Compléter le code suivant:

```
class CostumeHalloween {
public:
        CostumeHalloween(const string& id) : id_(id) {}
        string getId() const { return id_; }
        void operator++() { nbFoisLoue_++; }

private:
        string id_;
        unsigned int nbFoisLoue_;
};
class MagasinCostumesHalloween {
public:
        const CostumeHalloween* louerCostume(const string& id) {
                for([unique_ptr<CostumeHalloween>&] costume: marchandises_){
                        if ([costume->]getId() == id) {
                                ++[(*costume)];
                                return [costume.get()];
                        }
                }
                return nullptr;
        }

private:
        vector<unique_ptr<CostumeHalloween>> marchandises_;
};
```

Question 12

Non répondue

Noté sur 1,00

Soit le code suivant:

```
1 int main(){
2     shared_ptr<string> ptr = make_shared<string>("pointeur");
3     shared_ptr<string> ptr_copie = move(ptr);
4     //Autres instructions du main
5 }
```

Vrai ou faux. Suite à l'exécution de la ligne 3, le pointeur ptr pourra être déréférencé afin d'afficher le contenu de sa ressource.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Faux ».

Question 13

Non répondue

Noté sur 3,00

Soit le programme suivant:

```
void uneFonction(shared_ptr<string> ptr) {
        cout << ptr.use_count() << endl; // A
}

int main() {
        shared_ptr<string> ptr = make_shared<string>("pointeur");
        cout << ptr.use_count() << endl; //B
        uneFonction(move(ptr));
        cout << ptr.use_count() << endl; //C
}
```

Pour chaque affichage, dites quel sera le compte de référence.

Affichage A Choisir…

Affichage B Choisir…

Affichage C Choisir…

Votre réponse est incorrecte.

La réponse correcte est : Affichage A → 1, Affichage B → 1,
Affichage C → 0

Description

Soit la hiérarchie des classes suivantes:

```
class EnLigne{
public:
   virtual ~EnLigne()= default;
   virtual void setZoom()= 0;
};
class Cours {
public:
  virtual ~Cours()= default;
  void setResponsable();
  virtual void setCoursDisponible()= 0;
  friend ostream & operator << (ostream  &  out,  const  Cours & c);
protected :
  string  titre_;
 };
class CoursCycleSuperieur: public Cours{
public:
  virtual ~CoursCycleSuperieur()= default;
  int getTitre();
  void setCoursDisponible() override;
  virtual void setRequis();
  friend ostream & operator << (ostream  &  out,  const  CoursCycleSuperieur & c);
private:
  string preRequis_;
};

class CoursPremierCycle: public Cours{
public:
  virtual ~CoursPremierCycle()= default;
  int getTitre();
  void setCoursDisponible() override;
  friend ostream & operator << (ostream  &  out,  const  CoursPremierCycle & c);
private:
  string coursPrerequis_;
 };

 class CoursPremierCycleEnLigne: public CoursPremierCycle,public EnLigne
 { public:
    virtual ~CoursPremierCycleEnLigne()= default;
    void setNombreEtudiant();
    void setZoom() override;
 };
;
class Cours8000: public CoursPremierCycle, public CoursCycleSuperieur{
  public:
    virtual ~Cours8000()= default;
    void afficher();
    void setCoursDisponible()override;
    void setRequis()override;
};
```

Question 14

Non répondue

Noté sur 6,00

Pour chaque classe indiquer la forme de la classe ?

CoursPremierCycleEnLigne	Choisir…
EnLigne	Choisir…
CoursCycleSuperieur	Choisir…
Cours8000	Choisir…
Cours	Choisir…
CoursPermierCycle	Choisir…

Votre réponse est incorrecte.

La réponse correcte est :
CoursPremierCycleEnLigne → Concrete, EnLigne → Interface,

CoursCycleSuperieur → Concrete,

Cours8000 → Concrete,

Cours → Abstraite,

CoursPermierCycle → Concrete

Question 15

Non répondue

Noté sur 8,00

Soient la déclaration et les appels des méthodes, identifier quelle méthode est effectivement appelée.

```
shared_ptr <Cours> unCours = make_shared<CoursCycleSuperieur>();
unCours->setCoursDisponible();
unCours->setRequis();
```

```
shared_ptr <Cours> unCours = make_shared<CoursPremierCycle>();
unCours->setResponsable();
dynamic_cast<CoursPremierCycle*>(unCours.get())->getTitre();
```

```
shared_ptr <CoursPremierCycleEnLigne> unCours = make_shared<CoursPremierCycleEnLigne>();
unCours->setZoom();
unCours->setNombreEtudiant();
```

```
shared_ptr <CoursCycleSuperieur> unCours = make_shared<Cours8000>();
unCours->getTitre();
unCours->setRequis();
```

Votre réponse est incorrecte.

La réponse correcte est : Soient la déclaration et les appels des méthodes, identifier quelle méthode est effectivement appelée.

```
shared_ptr <Cours> unCours = make_shared<CoursCycleSuperieur>();
unCours->setCoursDisponible();[CoursCycleSuperieur::setCoursDisponible()]
unCours->setRequis();[Aucune]
```

```
shared_ptr <Cours> unCours = make_shared<CoursPremierCycle>();
unCours->setResponsable();[Cours::setResponsable()]
dynamic_cast<CoursPremierCycle*>(unCours.get())->getTitre();[CoursPremierCycle::getTitre()]
```

```
shared_ptr <CoursPremierCycleEnLigne> unCours = make_shared<CoursPremierCycleEnLigne>();
unCours->setZoom();[CoursPremierCycleEnLigne::setZoom()]
unCours->setNombreEtudiant();[CoursPremierCycleEnLigne::setNombreEtudiant()]
```

```
shared_ptr <CoursCycleSuperieur> unCours = make_shared<Cours8000>();
unCours->getTitre();[CoursCycleSuperieur::getTitre()]
unCours->setRequis();[Cours8000::setRequis()]
```

Question 16

Non répondue

Noté sur 4,00

Pour chaque appel d'une méthode, indiquer le choix ?

`Cours unCours; unCours.setResponsable();`	Choisir…
`Cours8000 unCours; unCours.getTitre();`	Choisir…
`Cours8000 unCours; unCours.afficher();`	Choisir…
`Cours8000 unCours; unCours.setResponsable();`	Choisir…

Votre réponse est incorrecte.

La réponse correcte est :

```
Cours unCours;
unCours.setResponsable();
```

→ L'objet unCours ne peut pas être instancié.,

```
Cours8000 unCours;
unCours.getTitre();
```

→ La méthode appelée est ambigüe puisqu'elle est définie dans plusieurs classes de base.,

```
Cours8000 unCours;
unCours.afficher();
```

→ Aucun problème de compilation.,

```
Cours8000 unCours;
unCours.setResponsable();
```

→ La méthode appelée est liée au problème du diamant.

Question 17

Non répondue

Noté sur 2,00

Dans la classe Cours et sa classe dérivée CoursPremierCycle, la fonction globale operator <<est amie. Comment peut-on appeler la fonction globale amie operator << de la classe de base dans la fonction globale operator << de la classe dérivée?

Veuillez choisir une réponse.

1. operator << (c);
2. out << c;
3. out << static_cast<const Cours &>(c);
4. Cours::operator << (c);
5. out << static_cast<Cours &>(c);
6. impossible

Votre réponse est incorrecte.

La réponse correcte est :
out << static_cast<const Cours &>(c);

Description

On vous demande de créer un programme qui gère une maison hantée par plusieurs forces surnaturelles. Dans les questions suivantes, vous devrez faire l'implémentation de plusieurs méthodes en faisant comme si vous étiez dans le .cpp des classes concernées. De plus, prenez pour acquis que toutes les inclusions nécessaires pour vos implémentations ainsi que le using namespace std ont déjà été ajoutés pour vous dans le code, nous n'avez donc pas à vous en soucier.

Question 18

Non répondue

Note de 0,00 sur 5,00

Modifier les classes suivantes pour rendre possible le polymorphisme sur les méthodes copie() et faireApparition(). Noter qu'il est impossible d'instancier un objet de la classe ForceSurnaturelle puisqu'elle est abstraite.

Réponse : (régime de pénalités : 0. %)

Réinitialiser la réponse

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

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom);
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        ForceSurnaturelle* copie() const {}
        void faireApparition() const {}
private:
        string nom_;
};

class Revenant : public ForceSurnaturelle {
public:
        Revenant(const string& nom);
        ForceSurnaturelle* copie() const { return new Revenant(*this); }
        void faireApparition() const { cout << "Forme humaine etrange"; }
```

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

Question 19

Non répondue

Note de 0,00 sur 5,00

Soient les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante{
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        void genererApparition() const {
                int fIndex = rand() % (fantomes_.size());
                fantomes_[fIndex]->faireApparition();
        }
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        void genererApparition() const;
};
```

Notez que ForceSurnaturelle est une classe abstraite. Faites l'implémentation du constructeur par paramètres de la classe MaisonHante.

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 20

Non répondue

Note de 0,00 sur 3,00

Soit les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante{
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        void genererApparition() const {
                int fIndex = rand() % (fantomes_.size());
                fantomes_[fIndex]->faireApparition();
        }
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        void genererApparition() const;
};
```

Notez que ForceSurnaturelle est une classe abstraite. Faites l'implémentation du constructeur par copie de la classe MaisonHante. On cherche à produire une copie exacte.

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 21

Non répondue

Note de 0,00 sur 8,00

Soient les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante{
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        void genererApparition() const {
                int fIndex = rand() % (fantomes_.size());
                fantomes_[fIndex]->faireApparition();
        }
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        void genererApparition() const;
};
```

Notez que ForceSurnaturelle est une classe abstraite. Faites l'implémentation de la surcharge de l'opérateur d'affection de la classe LieuHante. On cherche à produire une copie exacte.

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 22

Non répondue

Note de 0,00 sur 12,00

Soient les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante{
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        void genererApparition() const {
                int fIndex = rand() % (fantomes_.size());
                fantomes_[fIndex]->faireApparition();
        }
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        void genererApparition() const;
};
```

Notez que ForceSurnaturelle est une classe abstraite et que la classe MaisonHante hérite de deux classes. Faites l'implémentation de la surcharge de l'opérateur d'affection de la classe MaisonHante. On cherche à produire une copie exacte.

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 8,00

Soient les classes suivantes:

```
class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};
```

Faites l'implémentation de la surcharge de l'opérateur -= de la classe Maison qui retire du vecteur residents_ la personne passée en paramètre à la méthode seulement si elle est présente dans le vecteur.

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 5,00

Soient les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante{
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        void genererApparition() const {
                int fIndex = rand() % (fantomes_.size());
                fantomes_[fIndex]->faireApparition();
        }
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        void genererApparition() const;
};
```

Notez que ForceSurnaturelle est une classe abstraite. Complétez l'implémentation de la méthode MaisonHante::genererApparition() qui génère une apparition d'une force surnaturelle d'un des fantômes de la maison, choisi aléatoirement, dans une pièce donnée, aussi choisie aléatoirement.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

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

```
void MaisonHante::genererApparition() const {
        /*TODO 1: Générer l'apparition d'une ForceSurnaturelle*/
        int pIndex = rand() % (getNbPieces() - 1);
        cout << " dans " << /*TODO 2: Afficher la piece correspondante à pIndex dans Maison::pieces_*/
<< endl;
}
```

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

Question 25

Non répondue

Noté sur 4,00

Soient les classes suivantes:

```
class ForceSurnaturelle {
public:
        ForceSurnaturelle(const string& nom) : nom_(nom) {}
        bool operator==(const ForceSurnaturelle& force) const {
                return force.nom_ == nom_;
        }

        /*ForceSurnaturelle* copie()*/
        /*void faireApparition()*/
private:
        string nom_;
};
class LieuHante {
public:
        virtual ~LieuHante() = default;
        LieuHante(const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        LieuHante& operator+=(const ForceSurnaturelle& force);
        LieuHante& operator-=(const ForceSurnaturelle& force);
        ForceSurnaturelle& operator[](int index);

        string getAdresse() const;
        int getNbFantomes() const;

        /*genererApparition()*/
private:
        string adresse_;
        vector<unique_ptr<ForceSurnaturelle>> fantomes_;
};

class Personne {
public:
        Personne(string nom = "", int age = 0) : nom_(nom), age_(age) {}

        bool operator==(const Personne& personne) const {
                return personne.nom_ == nom_;
        }
private:
        string nom_;
        int age_;
};

class Maison {
public:
        Maison(int nbPieces, string pieces[], const string& adresse = "");
        /*Constructeur de copie et opérateur d'affectation*/

        Maison& operator+=(const shared_ptr<Personne>& personne);
        Maison& operator-=(const shared_ptr<Personne>& personne);
        string& operator[](int index);

        string getAdresse() const;
        int getNbPieces() const;
        int getNbResidents() const;

private:
        string adresse_;
        int nbPieces_;
        unique_ptr<string[]> pieces_;
        vector<shared_ptr<Personne>> residents_;
};

class MaisonHante : public Maison, public LieuHante {
public:
        virtual ~MaisonHante() = default;
        MaisonHante(int nbPieces, string pieces[], const string& adresse);
        /*Constructeur de copie et opérateur d'affectation*/

        /*genererApparition()*/
};
```

Dans le programme que vous venez de créer, MaisonHante est en relation d'héritage avec Maison ET LieuHante. Ainsi, certaines méthodes créeront éventuellement des problèmes lors de leur appel. Dites quelle(s) méthode(s) pourraient devenir problématique(s) et dites comment le code devrait être modifié afin de résourdre ce(s) probléme(s).