Question **1**

Correct

Note de 1,00 sur 1,00

```
list<int> v;
auto it = v.begin();
for (int i : range(n)) {
    v.insert(it, f(i));
    --it;
    v.insert(it, f(-i));
}
```

Est $O($ n ✔ $)$

Votre réponse est correcte.

La réponse correcte est :

```
list<int> v;
auto it = v.begin();
for (int i : range(n)) {
    v.insert(it, f(i));
    --it;
    v.insert(it, f(-i));
}
```

Est $O([n])$

Question **2**

Correct

Note de 1,00 sur 1,00

```
vector<int> v;
for (int i : range(n))
    for (int j : range(n))
        v.push_back(f(i,j));
```

Est $O($ n^2 ✔ $)$

Votre réponse est correcte.

La réponse correcte est :

Question **3**

Incorrect

Note de 0,00 sur 1,00

```
deque<int> v;
for (int i : range(n))
    v.insert(v.begin()+v.size()/2, f(i));
```

Est $O($ $n \cdot \log(n)$ ✖)

Votre réponse est incorrecte.

La réponse correcte est :

```
deque<int> v;
for (int i : range(n))
    v.insert(v.begin()+v.size()/2, f(i));
```

Est $O([n^2])$

Question **4**

Partiellement correct

Note de 0,50 sur 1,00

Choisissez dans la liste **TOUTES** les affirmations **VRAIES** au sujet de l'architecture modèle-vue-controlleur

- [x] a. Le contrôleur doit vérifier la validité des actions de l'utilisateur. ✔
- [x] b. La vue doit continuellement vérifier l'état du modèle en cas de changement. ✖
- [ ] c. Toutes les mises à jour de la vue doivent passer par le modèle.

Votre réponse est partiellement correcte.

Vous avez sélectionné trop d'options.

La réponse correcte est :

Le contrôleur doit vérifier la validité des actions de l'utilisateur.

```cpp
struct ErreurCoursEnLigne : std::logic_error {
    using logic_error::logic_error;
};

struct ErreurIntraTropDifficile : ErreurCoursEnLigne {
    using ErreurCoursEnLigne::ErreurCoursEnLigne;
};

struct ErreurRienCompris : ErreurIntraTropDifficile {
    using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
};

struct ErreurPasAssezÉtudié : ErreurIntraTropDifficile {
    using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
};

struct ErreurTwitch : ErreurCoursEnLigne {
    using ErreurCoursEnLigne::ErreurCoursEnLigne;
};

struct ErreurColocsDuChargéOntRebranchéLeRouteur : ErreurTwitch {
    using ErreurTwitch::ErreurTwitch;
};


class Étudiant {
public:
    void étudier(bool pourVrai) {
        if (pourVrai) {
            // Refaire ses TD par soi-même et refaire les exemples faits en classe. :D
            prêtPourExamen_ = true;
        } else {
            // Meh, relire les notes de cours une fois la veille de l'exam. :(
            prêtPourExamen_ = false;
        }
    }

    void assisterAuCours() {
        // Assister au stream sur Twitch.
        bool coursFini = false;
        while (not coursFini) {
            if (not streamTwitchActif())
                throw ErreurColocsDuChargéOntRebranchéLeRouteur("Ah come on!");

            coursFini = /* Est-ce que le chargé a fini de monologuer et/ou commence à avoir faim? */;
        }
    }

    void faireIntra() {
        double note = /* Est-ce que ça c'est bien passé? */;
        if ( note < 10/20.0 ) {
            if (not prêtPourExamen_)
                throw ErreurPasAssezÉtudié("Oups! Les questions demandaient de comprendre...");
            else
                throw ErreurRienCompris("De toute façon, les examens sont juste des constructions sociales.");
        } else if (note <= 15/20.0) {
            cout << "Pas pire, pas pire." << "\n";
        } else if (note <= 20/20.0) {
            cout << "Yay!" << "\n";
        }
    }

private:
    bool prêtPourExamen_;
};
```

Question **5**

Correct

Note de 2,00 sur 2,00

Soit le code suivant :

```
1. int main () {
2.     Étudiant steveFleury;
3.     try {
4.         steveFleury.assisterAuCours();
5.         steveFleury.étudier(false);
6.         steveFleury.faireIntra();
7.     } catch (ErreurPasAssezÉtudié& e) {
8.         // Catch 1
9.     } catch (ErreurRienCompris& e) {
10.        // Catch 2
11.    } catch (ErreurIntraTropDifficile& e) {
12.        // Catch 3
13.    } catch (ErreurCoursEnLigne& e) {
14.        // Catch 4
15.    }
16.
17.    // Fin du programme
18. }
```

Qu'arrive-t-il si le stream est interrompu (`streamTwitchActif()` retourne faux) et qu'on présume que l'élève obtiendrait une note de 8/20?

Veuillez choisir une réponse :

- a. Une exception est levée dans `faireIntra()`, le catch 2 est exécuté, puis le programme se termine normalement.
- b. Une exception est levée dans `assisterAuCours()`, aucun catch ne peut attraper le `ErreurColocsDuChargéOntRebranchéLeRouteur` et le programme plante à cause de l'exception non gérée.
- c. Une exception est levée dans `assisterAuCours()`, le catch 4 est exécuté et le programme se termine normalement. ✔

Votre réponse est correcte.

La réponse correcte est : Une exception est levée dans `assisterAuCours()`, le catch 4 est exécuté et le programme se termine normalement.

Question **6**

Correct

Note de 1,00 sur 1,00

Choisissez dans la liste **TOUTES** les bonnes raisons d'attraper les exceptions par référence dans les blocs `try-catch`

Veuillez choisir au moins une réponse :

- [ ] a. Pour pouvoir utiliser les méthodes statiques des objets d'exception.
- [ ] b. Pour utiliser les move semantics.
- [x] c. Pour garder le polymorphisme des exceptions lancées. ✔
- [x] d. Pour éviter les copies inutiles d'objets d'exception. ✔

Votre réponse est correcte.

Les réponses correctes sont : Pour garder le polymorphisme des exceptions lancées., Pour éviter les copies inutiles d'objets d'exception.

Soit les définitions de classes suivantes (dans des fichiers séparés et on suppose que les "include" sont faits correctement compile):

```
class A : public B, public C {
public:
  A() { }
private:
  C att1_;
  B* att2_;
};

class B {
public:
  B() { }
};

class C {
public:
  C() { }
};

class D : public C {
public:
  D() { att2_ = new A(); }
private:
  B att1_;
  A* att2_;
};
```

Soit le programme suivant:

```
int main() {
  D x;
}
```

On veut savoir l'ordre de construction lorsqu'on exécute ce programme. Indice: il y a 7 objets/sous-objets construits.

Indiquez uniquement les noms des types dans le bon ordre, exemple: A B C D A B C

Réponse : (régime de pénalités : 0 %)

Indiquez l'ordre de début de construction pour les classes ci-haut:

D C B A B C C

Indiquez l'ordre de début d'exécution du corps des constructeurs pour les classes ci-haut:

C B D B C C A

Question **8**

Correct

Note de 1,50 sur 1,50

Cette question devrait être faite sans utiliser un compilateur.

Soit le programme suivant. Dans chaque case écrivez une seule valeur ou expression de manière à ce que l'exécution du programme corresponde à l'affichage.

```
bool estInf(int i)
{
   return (i < 100 );
}
int main() {

    vector<int> v = { 100, 116, 125, 2, 47 };

   auto it = remove_if(v.begin(),v.end(), estInf);

   int i = 0;
   for (auto it = [v.begin()] ; it != [v.end()] ; ++it)
   {
      cout << *it << " ";
      *it = ++i;
   }
}
```

Affichage :

100 116 125 [2] [47]

Nous avons besoin d'une classe **Cryptomonnaie** qui servira comme classe de base pour plusieurs cryptomonnaies.

Cette classe contiendra les élements suivants:
- Un constructeur et un destructeur par défaut.
- Un attribut **valeur_** de type **réel à double précision** qui représente la valeur d'une unité de la monnaie en dollars.
- Les méthodes non virtuelles **getValeur** et **changerValeur** qui permettent d'obtenir et de modifier la valeur_.
- Une méthode virtuelle pure **obtenirSymbole** qui retourne le symbole de la monnaie sous forme d'une **chaine de caractèr**
- Une méthode virtuelle pure **estMinable** qui retourne un **booléen** indiquant s'il est possible de miner cette cryptomonnaie

Donnez la définition de cette classe. Si des méthode sont à définir, vous pouvez le faire dans ou à l'extérieur de la classe, vous voulez.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
class Cryptomonnaie {
    //TODO: Définition de la classe
public:
    virtual ~Cryptomonnaie() = default;
    virtual string obtenirSymbole() const = 0;
    virtual bool estMinable() const = 0;
    double getValeur() const { return valeur_; };
    void changerValeur(double nouvelleValeur) { valeur_ = nouvelleValeur; }

private:
    double valeur_;
};
```

	Test	Résultat attendu	Résultat obtenu	
✔	_			✔
✔	`cout << is_abstract_v<Cryptomonnaie> << endl;`	1	1	✔
✔	`// L'objet a la bonne taille`	1	1	✔
✔	`unique_ptr<Cryptomonnaie> crypto;` `cout << permet_unique_ptr_Crypto;`	1	1	✔
✔	`// test non montré`			✔
✔	`// test non montré`			✔

✔	`// Réussi à instancier une classe dérivée et`
	`cout << has_virtual_obtenirSymbole_v<Cryptomonnaie>;`
	`cout << has_virtual_estMinable_v<Cryptomonnaie>;`
		11	11	✔

Correct

Note de 1,25 sur 1,25

Nous voulons utiliser la classe précédente afin de définir une nouvelle classe Dogecoin qui héritera de Cryptomonnaie.

Donnez la définition et l'implémentation de la classe Dogecoin tel que:
- Le symbole de la monnaie Dogecoin est "D".
- Il est possible de miner la monnaie Dogecoin.
- Dogecoin n'est pas une classe abstraite.

L'implémentation des méthodes peut être à l'intérieur ou à l'extérieur de la classe.

NOTE: Cette question n'utilise pas votre version de la classe précédente et peut donc être répondue sans avoir réussi l'autre question.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
class Dogecoin : public Cryptomonnaie {
    // TODO: Définition de la classe
public:
    virtual bool estMinable() const override { return true; };
    virtual string obtenirSymbole() const override { return symbole; };

private:
    string symbole = "D";
};

// TODO: Implémentation des méthodes (peut aussi être dans la classe)
```

	Test	Résultat attendu	Résultat obtenu	
✔	_			✔
✔	`Dogecoin doge;` `cout << is_base_of_v<Cryptomonnaie, Dogecoin>;`	true	true	✔

✔	`Dogecoin* d = nullptr;`
	`Cryptomonnaie* c = d;`
	`cout << "ok";`
		ok	ok	✔

✔	`Dogecoin doge;`
	`Cryptomonnaie& cryp = doge;`
	`cout << doge.obtenirSymbole() << "\n";`
		D	D	✔

✔	`Dogecoin doge;`
	`Cryptomonnaie& cryp = doge;`
	`cout << doge.estMinable() << "\n";`
		true	true	✔

✔	`// test non montré`	true	true	✔

Tous les tests ont été réussis ! ✔

Description

Les questions de cette page utilisent la classe Cryptomonnaie de la page précédente; vous pouvez l'ouvrir dans un autre onglet de votre navigateur pour y accéder facilement.

Afin de garder l'information sur les cryptomonnaies détenues par un individu, nous définissons la classe suivante:

```
class Portefeuille {
public:
    Portefeuille();
    ~Portefeuille();

    double valeurEnDollars(const std::map<std::string, Cryptomonnaie*> cryptos) const;

private:
    std::map<std::string, double> quantites_;
};
```

L'attribut **quantites_** est une map qui contient les quantités d'unités détenues de chaque cryptomonnaie. Les clés de la map sont les symboles des monnaies et la valeur est la quantité d'unités de la monnaie dans le portefeuille.

La méthode **valeurEnDollars** reçoit en paramètre une autre map qui contient les symboles des monnaies comme clés et des pointeurs vers les définitions de monnaies comme valeurs (par exemple, la clé "D" aura comme valeur un pointeur vers un objet de type Dogecoin).

En utilisant des algorithmes et des fonctions lambdas, nous voulons implémenter la méthode **valeurEnDollars** qui retourne la valeur totale du portefeuille en dollars.
Cette question se décompose en plusieurs étapes.

Question **11**

Correct

Note de 1,00 sur 1,00

Écrire une fonction lambda qui retourne la somme de deux réels double reçus en paramètres.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```cpp
std::function<double(double, double)> lambdaAddition()
{
    return [](double x, double y) { return x + y; };
}
```

	Test	Résultat attendu	Résultat obtenu	

✔	`auto fn = lambdaAddition();`
	`double result = fn(42.2, 69.3);`
	`checkResult(result, 111.5);`
	`result = fn(12.5, 7.25);`
	`checkResult(result, 19.75);`
		111.5
		19.75
			111.5
			19.75
				✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Cpp):

```cpp
std::function<double(double, double)> lambdaAddition()
{
    return [](auto a, auto b) { return a + b; };
}
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **12**

Correct

Note de 1,50 sur 1,50

Écrire une fonction lambda capable de calculer la valeur d'une quantité de monnaie d'un certain type.

La fonction lambda reçoit comme paramètre une paire de <string, double> qui contient le symbole d'une monnaie et sa quantité en unités.
La fonction lambda peut aussi avoir accès à une map qui contient les symboles des monnaies comme clés et des pointeurs vers les définitions de monnaies comme valeurs (par exemple, la clé "D" aura comme valeur un pointeur vers un objet de type Dogecoin).

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
using TypeLambda = function<double(const pair<string, double>&)>;
// Ne pas modifier le nom/signature de la fonction lambdaEvaluerQuantite
TypeLambda lambdaEvaluerQuantite(map<string, Cryptomonnaie*>& cryptos)
{
    // "cryptos" contient les symboles des monnaies comme clés et des pointeurs vers les définitions de
    return [&](pair<string, double> p) { return p.second * cryptos[p.first]->getValeur(); };
}
```

	Test	Résultat attendu	Résultat obtenu	

✔	`MaCrypto maCrypto; maCrypto.changerValeur(42);`
	`map<string, Cryptomonnaie*> cryptos = {{"M"s, &maCrypto}};`
	`auto fn = lambdaEvaluerQuantite(cryptos);`
	`cout << fn({"M"s, 2.0}) << "\n"`
	`  << fn({"M"s, 10.0}) << "\n";`
		84
		420
			84
			420
				✔

✔	`MaCrypto maCrypto1; maCrypto1.changerValeur(42);`
	`MaCrypto maCrypto2; maCrypto2.changerValeur(69);`
	`map<string, Cryptomonnaie*> cryptos1 = {{"M", &maCrypto1}};`
	`map<string, Cryptomonnaie*> cryptos2 = {{"M", &maCrypto2}};`
	`auto fn1 = lambdaEvaluerQuantite(cryptos1);`
	`auto fn2 = lambdaEvaluerQuantite(cryptos2);`
	`cout << fn1({"M"s, 2.0}) << ", " << fn1({"M"s, 10.0}) << "\n"`
	`  << fn2({"M"s, 2.0}) << ", " << fn2({"M"s, 10.0}) << "\n";`
		84, 420
		138, 690
			84, 420
			138, 690
				✔

Tous les tests ont été réussis ! ✔

Question **13**

Incorrect

Note de 0,00 sur 2,00

Maintenant que nous avons une fonction lambda pour calculer la somme de deux réels et une fonction lambda capable de calculer la valeur d'une quantité de cryptomonnaie donnée, nous pouvons facilement calculer la valeur totale d'un portefeuille de cryptomonnaies.

Implémentez la méthode **valeurEnDollars** de la classe Portefeuille en utilisant les deux fonctions lambda définies précédemment ainsi que l'algorithme **transform_reduce**. On utilise la version qui prend un foncteur binaire et un foncteur unaire (la version vue dans les notes de cours, et la version 3 dans la doc sur cppreference). Vous n'avez évidemment pas le droit d'utiliser **for** ou **while** à la place.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  double Portefeuille::valeurEnDollars(const map<string, Cryptomonnaie*>& cryptos) const
2  {
3      // Récupération de nos fonctions lambda
4      auto lambdaUn = lambdaAddition();
5      auto lambdaDeux = lambdaEvaluerQuantite(cryptos);
6
7      // TODO: Calculer la valeur totale du portefeuille à l'aide des fonctions lambda et de l'algorithme
8      return transform_reduce(cryptos.begin(), cryptos.end(), 0, lambdaUn, lambdaDeux);
9
10 }
```

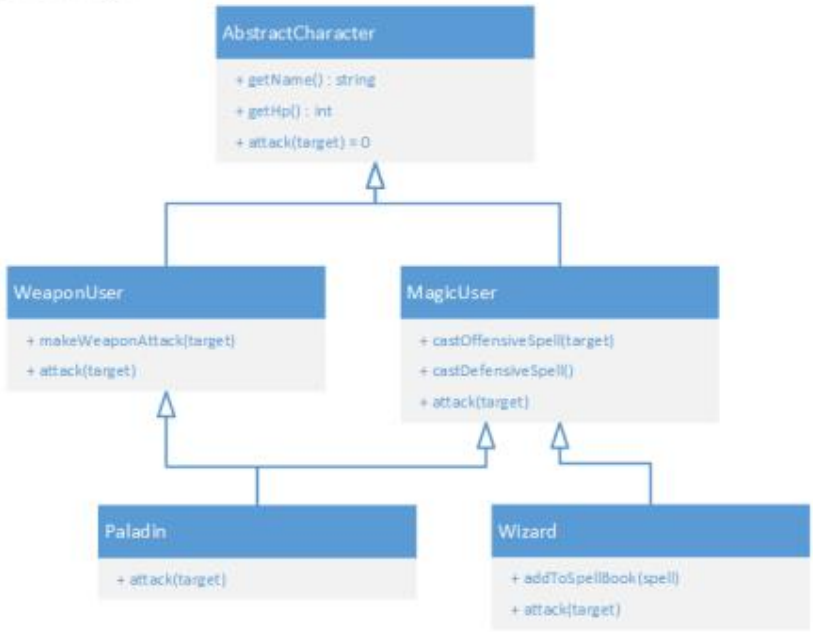
Erreur(s) de syntaxe

Question **14**

Correct

Note de 2,00 sur 2,00

Nous avons la hiérarchie de classes suivante qui représente les personnages dans un jeu de fantaisie. Dans ce jeu, un sorcier (classe `Wizard`) est un personnage qui utilise la magie et un paladin (classe `Paladin`) est un personnage qui peut utiliser autant des armes que de la magie.



On voit facilement un problème d'héritage en diamant pour `Paladin`. Dans le code ci-dessous, remplissez les boîtes avec le type d'héritage approprié pour que les classes soient compilables et utilisables. Si vous avez à le faire, mettez le moins d'héritage virtuel possible.

```cpp
class AbstractCharacter {
public:
    AbstractCharacter(const string& name, int hp)
    : name_(name), hp_(hp) { }

    virtual ~AbstractCharacter() = default;

    const string& getName() const { return name_; }
    int getHp() const { return hp_; }

    virtual void attack(AbstractCharacter& target) = 0;

protected:
    string name_;
    int hp_;
};

class WeaponUser : public virtual AbstractCharacter {
public:
    WeaponUser(const string& name, int hp, Weapon* weapon)
    : AbstractCharacter(name, hp),
      weapon_(weapon) { }

    void makeWeaponAttack(AbstractCharacter& target) {
        // On attaque avec notre arme...
    }

    void attack(AbstractCharacter& target) override {
        makeWeaponAttack(target);
    }

protected:
    Weapon* weapon_;
};

class MagicUser : public virtual AbstractCharacter {
public:
    MagicUser(const string& name, int hp, const vector<Spell*>& spellList)
    : AbstractCharacter(name, hp),
      spellList_(spellList) { }

    void castOffensiveSpell(AbstractCharacter& target) {
        // On attaque avec un sort...
    }

    void castDefensiveSpell() {
        // On se protège avec de la magie...
    }

    void attack(AbstractCharacter& target) override {
        castOffensiveSpell(target);
    }

protected:
    vector<Spell*> spellList_;
};

class Wizard : public MagicUser {
public:
    Wizard(const string& name, int hp, const vector<Spell*>& spellList)
    : AbstractCharacter(name, hp),
      MagicUser(name, hp, spellList) { }

    void addToSpellBook(Spell* spell) {
        spellList_.push_back(spell);
    }
};

class Paladin : public WeaponUser, public MagicUser {
public:
    Paladin(const string& name, int hp, Weapon* weapon, const vector<Spell*>& spellList)
    : AbstractCharacter(name, hp),
      WeaponUser(name, hp, weapon),
      MagicUser(name, hp, spellList) { }

    void attack(AbstractCharacter& target) override {
        WeaponUser::attack(target);
    }
};
```