1 . Quel est le rôle principal du constructeur d'une classe ?

◯ Allouer la mémoire nécessaire à une instance de cette classe

◯ Initialiser une instance de cette classe

◯ Renvoyer l'adresse de l'objet

◯ Défragmenter la mémoire

2 . Quel est le rôle principal du destructeur d'une classe ?

◯ Libérer la mémoire occupée par un objet

◯ Détruire un objet

◯ Libérer les ressources de l'objet

◯ Défragmenter la mémoire

3 . Dans quelle situation la zone d'initialisation des membres est-elle indispensable ?

◯ Pour initialiser des membres de type pointeur

◯ Pour initialiser les membres 'static'

◯ Pour initialiser les membres de type référence ou const

◯ Pour initialiser des membres de type tableau

4 . Identifiez l'assertion erronée au sujet des fonctions 'inline' :

◯ Elles peuvent contenir plusieurs instructions

◯ Elles doivent être 'static'

◯ Elles améliorent les performance du programme

◯ Elles sont expansées à l'endroit de chaque appel

5 . Quelle instruction permet de réémettre l'exception courante ?

◯ throw ; ◯ throw() ; ◯ throw() ; ◯ rethrow;

6 . Quel(s) opérateur(s) peut(peuvent) être surchargé(s) ?

◯ * ◯ [] ◯ () ◯ Tous

7 . Quelle instruction n'appelle PAS l'opérateur d'affectation de la classe A ? (avec A a1; et A& ra1 = a1;)

◯ A a = a1 ; ◯ a = a1 ; ◯ a.operator=(a1) ; ◯ ra1 = a;

8 . Combien d'appels au constructeur de copie sont-ils induits par le code suivant :

```
(.h) List reverse(const List l);
(.cpp) List listeA;
List listeB = reverse(listeA);
```

◯ 0 ◯ 1 ◯ 2 ◯ 3

9 . Que se passe-t'il si l'on ne définit pas de constructeur dans une classe ?

◯ La compilation échoue

◯ Un constructeur qui ne fait rien est généré par le compilateur

◯ Un constructeur qui initialise les membres à 0 (ou null) est généré par le compilateur

◯ Un constructeur qui lève une exception INVALID_CTOR est généré par le compilateur.

10 . Que se passe-t'il si l'on ne définit pas d'opérateur d'affectation dans une classe ?

◯ La compilation échoue

◯ Un opérateur = qui ne fait rien est généré par le compilateur

◯ Un operateur = qui fait une copie membre à membre est généré par le compilateur

◯ Un operateur = qui fait un appel au constructeur de copie est généré par le compilateur

◯ Un operateur = qui lève une exception INVALID_OPERATION est généré par le compilateur.

11 . Quelle est la signature du constructeur de copie d'une classe T ?

◯ const T& T::T(const T& original) { . . . }

◯ T& T::T(const T& original) { . . . }

◯ T T::T(T original) { . . . }

◯ T::T(const T original) { . . . }

◯ T::T(T original) { . . . }

◯ T::T(const T& original) { . . . }

12 . Soit le code suivant :

```
int i = 10;
const int IC = 20;
const int * pic;
```

Laquelle des affectations suivantes est illégale ?

◯ pic = &IC ; ◯ pic = &i ; ◯ *pic = IC ; ◯ i = *pic;

13 . Soit le code suivant :

```
int i = 10;
int j;
```

Laquelle des lignes suivantes est légale ?

◯ int * const pi = &j; *pi = i;

◯ int * const pi; pi = &j;

◯ int * const pi = &j; pi = &i;

14 . Soit une classe T et t un objet de type T. Quelle fonction faut-il implémenter pour pouvoir écrire le code suivant : cout << t;

◯ void T::operator << (ostream & os) { . . . }

◯ void operator << (const T& t, ostream & os) { . . . }

◯ ostream& T::operator << (ostream & os) { . . . }

◯ ostream& operator << (ostream & os, const T& t) { . . . }

15 . Soit le code suivant :

```
class Point
{
   private:
      int x, y;

   public:
      Point(int x, int y):x(x),y(y) { }
      int X() const { return x; } // (1)
      int Y() const { return y; } // (2)
      int& X() { return x; } // (3)
      int& Y() { return y; } // (4)
};

class ObjetGraphique
{
  public:
   ObjetGraphique(Point &p, int couleur) : point(p)
   {  this->couleur = couleur;   }
   void afficher() { cout << point.X() << "," << point.Y() << endl ; }

  private:
   int    couleur;
   const Point &point;
};

int main(int, char**)
{
   Point p1(0, 0);
   ObjetGraphique o1(p1, 1);

   o1.afficher(); // (5)
}
```

Lorsque la ligne (5) est exécutée, quelles méthodes sont appelées ?

◯ (1) et (2)

◯ (3) et (4)