Examen de Génie Logiciel et C++ ISIMA Tronc Commun 2ème Anné

Nom: **Prénom :** **Filière :**

Exemple de question : Quelle est la différence principale entre les mots clés struct et class ?

- ❑ Il n'y a aucune différence.
- ▪ Les attributs et fonctions membres d'instance d'un struct sont publiques par défaut.
- ❑ Les membres d'un struct sont privés par défaut.
- ❑ Il n'est pas possible de définir des membres privés pour un struct alors que pour une classe, c'est possible.

Question 1 : Quelle est la principale différence entre les fonctions inline et les macros #define ? (1 pt)

- ❑ Les macros #define sont définies dans un fichier d'en-tête mais les fonctions inline sont définies dans le fichier source.
- ▪ Contrairement aux macros #define, les fonctions inline évaluent chaque argument, contrôlent leur type et réalisent les conversions nécessaires.
- ❑ A l'exécution les macros #define permettent une plus grande rapidité.
- ❑ Il n'y a aucune différence.

Question 2 : Doit-on détruire explicitement les objets locaux ? Réponse :

- ❑ Tous les objets locaux doivent être détruits explicitement sauf les pointeurs.
- ❑ Oui, tout objet déclaré localement doit être détruit.
- ▪ Non, l'appel automatique du destructeur des objets locaux s'effectue au niveau de l'accolade fermante du bloc.
- • Seuls les pointeurs doivent être détruits explicitement.

Question 3 : qu'est-ce qu'une classe abstraite ?

- ❑ C'est une classe qui n'a aucune donnée membre.
- ❑ C'est une classe qui a toutes ses méthodes virtuelles.
- ▪ C'est une classe qui possède au moins une méthode virtuelle pure.
- ❑ C'est une classe qui a toutes ses données membres en privé.

Question 4 : Qu'est-ce que retourne l'instruction new int[10000] ?

- ▪ Un pointeur sur un entier (int *).
- ❑ Un entier (int).
- • Un pointeur indéfini (void*).
- ❑ Un pointeur de type char*.

Question 5 : l'opérateur new alloue de la mémoire ?

- ❑ Il alloue une zone mémoire nécessaire à stocker le pointeur retourné.
- ▪ Il alloue une zone mémoire correspondant à la taille occupée par le type concerné.
- ❑ Il alloue la zone mémoire nécessaire à stocker le pointeur retourné ainsi qu'une zone correspondant à la taille occupée par le type concerné.
- ❑ Il alloue une zone mémoire de taille fixe qui sera agrandie ultérieurement et automatiquement lorsque cela sera nécessaire.

Question 6 : Ecrire une fonction template « clone », qui accepte une référence constante « x » sur un **p2**

objet de type « T » et retourne un pointeur de type « T ». Le corps de la fonction retourne ce pointeur simplement en faisant une allocation dynamique avec appel au constructeur de la classe « T » qui utilise la référence x. Ecrire ensuite une fonction main qui déclare et initialise une variable entière i à 5, puis déclare un pointeur sur un entier pClone qui est initialisé en appelant la méthode clone précédente avec l'entier i, puis la fonction main détruit le clone pointé (1 pt).

```
template <class T>
T * clone (T & o)
{
   return new T(o);
}

int main(int, char**)
{
  int   i      = 5;
  int * pClone = clone(i);

  delete pClone;
}
```

Question 7 : Une classe A possède une méthode virtuelle draw() qui affiche "A", une sousclasse B hérite de A et redéfinit cette méthode draw() pour afficher "B". Le constructeur de la classe A appelle la méthode draw(). Pour la classe B nous avons le constructeur par défaut. Qu'affiche le code : `B unB;` **?**

- Il affiche "A".
- Il affiche "B".
- Il affiche "AB".

Question 8 : Une classe A possède comme attribut d'instance un pointeur vers des caractères qui doit être alloué pour instancier un objet de classe A. On suppose que le programmeur n'a pas implémenté la forme canonique de Coplien. Nous avons un constructeur par défaut qui effectue bien l'allocation mémoire et un destructeur qui fait le rendu mémoire. Nous avons aussi un constructeur qui accepte une chaîne de caractères. Efin, nous disposons également d'une méthode afficher dans la classe A qui affiche la chaîne de caractères. Discuter et commenter ce qui se passe dans le code suivant (2 pts) :

```
void examen()
{
        A       unA;            // Création d'un 1er objet A

        unA = A("La Chaîne");   // Création d'un 2ème obj A temporaire par constructeur
                                // Affectation bit à bit du 2ème objet vers le 1er
                                // car pas d'opérateur=. Le 1er pointeur est perdu
                                // destruction du deuxième objet temporaire lorsque
                                // le ';' est atteint => Rendu mémoire => le 1er obj
                                // pointe sur une zone mémoire rendue. Etat indéfini
        unA.afficher();         // Affichage ou plantage suivant contexte d'exécution.
}                               // Destruction du 1er objet local (2ème delete sur la
                                // même zone mémoire => Segmentation fault…
```

Question 9 : Dans le code suivant pourquoi avons nous deux opérateurs d'indexation ? (1 pt)

```
class Vecteur
{
      protected :
            int tab[3] ;
      public :
            Vecteur()                   { tab[0]=0 ; tab[1]=1 ; tab[2]=2 ;}
```

```
            int operator[](unsigned int index) const  { return tab[index] ; }
            int & operator[](unsigned int index)      { return tab[index] ; }
} ;
```

- Pour accéder au références sur des éléments du vecteur
- Pour accéder aux éléments du vecteur en lecture mais aussi en écriture
- Pour accéder au références sur des éléments du vecteur et aussi aux éléments constants.