

Partiel

Lundi 5 Novembre 2012

Tous documents autorises (sauf copie du voisin, appareillage electronique, etc). Les telephones portables, comme tout autre moyen de communication vers l'exterieur, doivent être eteints. Le temps a disposition est d'une heure. Motivez bien vos reponses.

1 Exercice [3 points] (statique *vs.* dynamique)

Etant donne une classe K et le code suivant :

```
class K{};
typedef K *PK;

void g(K **k) { delete *k; }

void f(K &k) {
    K *k1, k2(k), **k3;
    k1 = new K(k2);
    k3 = new PK;
    *k3 = new K(*k1);
    g(k3);
}

int main() {
    K k;
    f(k);
}
```

1. combien d'instances de K sont creees a l'execution de ce programme ?
2. combien d'instances sont detruites ?
3. completer la classe K de sorte que toutes les constructions et destructions apparaissent, *via* un affichage, a l'ecran.

2 Exercice [3 points] (const est mon ami)

Soit l'extrait de programme suivant :

```

int i(0);
const int *p = &i;
cout << "i=" << i << endl;
cout << "*p=" << *p << endl;
i = 100;
cout << "i=" << i << endl;
cout << "*p=" << *p << endl;

```

1. quels sont les a l chages produits par son execution ?
2. faites-vous une di erence entre *p et i ?
3. l'utilisation d'une reference `const int &r = i;` en lieu et place du pointeur p aurait-elle modi e le comportement ?

3 Exercice [2 points] (non-mutable)

Soit le code suivant :

```

class Etalon {
private:
    mutable int valeur;
public:
    Etalon(int valeur) { this->valeur = valeur; }
    int getValeur() const { valeur++; return valeur; }
};

int main() {
    const Etalon e(10);
    cout << e.getValeur() << endl;
    cout << e.getValeur() << endl;
    cout << e.getValeur() << endl;
    return 0;
}

```

1. quels sont les a l chages produits a l'execution de la fonction `main` ?
2. quelle di erence faites-vous entre un objet constant et un objet non-mutable ?

4 Exercice [2 points] (polymorphisme)

Supposons l'existence de la fonction suivante :

```

void supprimer(K k[],int n) {
    for (int i=0; i<n; i++) {
        delete k[i];
    }
}

```

```

        k[i] = 0;
    }
}

```

1. quelles precautions doit-on prendre lors de l'ecriture de la classe `K` ?
2. ecrire une classe `K` minimale et la specialiser en une sous-classe `L`, contenant toutes deux une(des) instruction(s) d'ajout permettant d'illustrer le bon fonctionnement de la fonction `supprimer`

5 Exercice [6 points] (UML)

Soit le diagramme UML suivant :

Ecrire en C++ les classes qui correspondent au diagramme precedent avec les accesseurs des attributs et des relations ainsi que les constructeurs et destructeurs adequats (on supposera que le nombre de pieces d'un immeuble est connu a sa construction).

6 Exercice [4 points] (factorisation UML/C++)

1. illustrer en UML la factorisation conceptuelle (interdit d'utiliser un exemple du cours)
2. illustrer en C++ la factorisation d'implementation (interdit d'utiliser un exemple du cours)