EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours d'Informatique à la section SV
Sam J.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROGRAMMATION II

Corrigé Examen(et barème)

Exercice 1 : Questions sur les concepts (28 points)

1. (a) le constructeur de `B` à la ligne `13` tente d'initialiser une attribut qui ne lui appartient pas spécifiquement (dans sa section `:`). L'attribut `a` hérité de `A` ne doit pas être initialisé de cette façon mais par le constructeur de `A`.

   (b) le fait qu'il n'existe pas de constructeur par défaut dans la classe `A`. En effet, comme aucun constructeur explicite de la super-classe n'est appelé, c'est le constructeur par défaut qui l'est. La classe `A` n'en possède pas car le constructeur par défaut par défaut n'existe plus dès qu'un constructeur explicite est présent.

   (c) à la ligne `13`, `a(unA)` doit être remplacé par `A(unA)`.

2. L'opérateur `-` est un opérateur binaire (il requiert donc 2 opérandes). Comme il est défini comme opérateur interne son premier opérande est forcément `this`. Il ne reste plus donc de place que pour une seul opérande en argument.

   L'erreur est donc que `operator-` a un argument en trop dans son prototype. Pour corriger l'erreur, il suffit de supprimer l'un des arguments. Une autre façon de faire est de déclarer l'opérateur comme externe.

3. (a) le programme affiche :

   ```
   <<
   3
    >>
   ```

   la méthode `m` de la hiérarchie est virtuelle et elle s'applique à une référérence (le paramètre de la fonction `m`). Il y a donc résolution dynamique des liens.

   (b) — Si on supprime le mot-clé `virtual` de la classe `X` le programme affiche

   ```
   <<
   3
   ```

   Comme la méthode `m` n'est plus virtuelle, la résolution statique des liens a lieu (pas de polymorphisme).

   — Si on supprime le mot-clé `virtual` de la classe `Y` le programme affiche

   ```
   <<
   3
    >>
   ```

   car il suffit de déclarer la virtualité au plus haut de la hiérarchie.

   — si on supprime le mot-clé `const` dans la classe `X` le programme ne compile pas. Le paramètre de la fonction `m` est `const`. On ne peut donc pas lui appliquer une méthode non `const`.

   — si on supprime le mot-clé `const` dans la classe `Y` le programme compile. La méthode `m` de `Y` étant non const elle n'est pas considérée comme une redéfinition de la méthode héritée de `X`: la résolution dynamique des liens n'a pas lieu et le programme affiche:

   ```
   <<
   3
   ```

4. Dans les situations où l'on souhaite qu'une classe soit abstraite (non instanciable) sans qu'il y ait d'autres méthodes virtuelles pures à y placer naturellement. Par exemple, la super-classe `Collider` de votre projet aurait pu être définie avec une position et un rayon. Elle aurait alors pu être instantiable. Comme elle ne sert qu'à caractériser la forme des objets pour réaliser des tests de collision, elle n'a pas vraiment de raison d'être instanciée en tant que telle, donc on aurait pu alors y déclarer un destructeur virtuel pur.

5. (a) Le programme affiche :

   ```
   10
   10
   10
   10
   10
   ```

   (b) Le problème est que le destructeur de `B` n'est jamais appelé. L'objet `a` du programme principal n'est donc que partiellement détruit et tous les objets alloués dynamiquement par le constructeur de `B` ne sont jamais désalloués (fuite de mémoire).

   (c) Pour rectifier ce problème il suffit de déclarer le destructeur de `A` comme virtuel.

   (d) Le `vector` de `B` contient `n` fois le même pointeur sur l'entier `10`. Le destructeur va donc essayer de désallouer `n` fois le même emplacement mémoire ce qui est erroné.

Exercice 12: Conception OO et programmation (57 points)

Il existe plusieurs façons de coder les détails des classes impliquées.
Notez que seuls les prototypes étaient demandés dans la partie 1.
Un corrigé possible est fourni dans les fichier sources `Simulation.cc`
Points : les points sont donnés en commentaire dans le fichier `simulation.cc`.

Exercice 3 : Déroulement de programme [15 points]

Le programme affiche:

```
1) 9
2) 20
3) 6
4) 20
5) 30
@@@@@
6) 20
7) 14
8) 4
9) 14
10) 21
@@@@@
11) 6
12) 14
13) 4
14) 14
15) 6
16) 14
17) 21
18) 21
@@@@@
19) 6
20) 15
21) 4
22) 15
23) 22
@@@@@
```

Pour `B`, il faut dire

- les valeurs sont `a=3`, `A::b=4` et `B::b=3` pour les attributs (appel au constructeur par défaut de `B`)
- appel de `B::m1` (donne `9`)
- appel de `B::m2` (donne `20=5*4`) `B` a deux attributs `b` et l'appel `getB()` retourne la valeur de celui hérité de `A`
- appel de `A::m1` car pointeur mais `m1` non virtuelle (donne 6=2*3)
- appel de `B::m2` car pointeur et `m2` virtuelle (donne 20)
- appel de `A::m3` car pas de `m3` dans `B` (utilise `A::b` et appelle `A::m1` et `B::m2`+car pointeur et `m2` virtuelle), donne (30 =4+6+20)

Pour `C`, il faut dire (comme pour `B`)

- les valeurs sont `a=2`, `A::b=3`, `B::b=3`, `c=5` pour les attributs (appel au constructeur par défaut de `C`)
- appel de `C::m1` (donne `20=10*2`)
- appel de `C::m2` (donne `14=3*3+5`) `C` a deux attributs `b` et l'appel `getB()` retourne la valeur de celui hérité de `A` car getB non virtuel)
- appel de `A::m1` car pointeur mais `m1` non virtuelle (donne 4=2*2)
- appel de `C::m2` car pointeur et `m2` virtuelle (donne 14)
- appel de `A::m3` car pas de `m3` dans `C` (utilise `A::b` et appelle `A::m1` et `B::m2`+car pointeur et `m2` virtuelle), donne (21 =3+4+14)

Pour `D`, il faut dire

- les valeurs sont `2` et `3` (appel au constructeur par défaut de `A` car classe virtuelle) et `5` pour `c` (constructeur par défaut de `C`)
- appel de `D::m1`, c-à-d `B::m1`, dans les deux cas (donne `6=3*2`)
- appel de `D::m2` (= `C::m2` qui donne `14=3*3+5`)
- appel de `A::m1` (donne 4 = 2*2)
- appel de `D::m2` (= `C::m2`) car `m2` virtuelle (donne 14)
- pour `pb` c'est `D::m1` et `D::m2` qui sont appelées car polymorphisme
- pour `pa->m3()` et `pb->m3` la même chose se produit. La méthode `m2` appelée est celle de `C` (donne 21)
- et pour la fin, le scénario est analogue au cas B (début en ligne 53) mais avec des valeurs d'attributs différents