Série 11: pointeur et allocation dynamique avec vector

Lien avec le MOOC Initiation à la Programmation (en C++)

- **Document Exercices semaine 7 du MOOC (vus en semaine 10)**

Exercice complémentaire (ExC)

ExC 8 : Réseau d'amis (niveau 2)

Soient les déclarations suivantes définissant une structure Personne qui permet de mémoriser un ensemble de liens vers d'autres structures Personnes pour représenter son groupe d'amis:

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

// prédéclaration nécessaire pour le typedef
struct Personne;

typedef vector<const Personne*> Liste_Personnes;

struct Personne
{
  string nom;
  Liste_Personnes amis;
};
```

a) proposer une fonction **afficher_ami ()** qui admet en paramètres une référence constante d'une Personne **pers** et qui affiche son nom suivi de la liste de ses amis. Par exemple :

```
Les amis de Alice sont :
…..
```

Pour tester cette fonction on peut créer quelques structures de type Personne puis leur ajouter manuellement des amis comme dans cet exemple :

```
Personne alice={"Alice",{}} ;
Personne bob={"Bob",{}} ;
…
alice.amis.push_back(&bob) ;

…
```

b) proposer une fonction **est_ami_avec()** qui admet en paramètres deux références constantes, une vers une Personne **pers** et une string **nom** et qui renvoie le booléen true dès qu'on trouve que la string est le nom d'une Personne appartenant à la liste d'amis de **pers**.

```
bool est_amis_avec(const Personne& pers, const string& nom) ;
```

Tester cette fonction comme à la question précédente.

c) Plutôt que d'ajouter les amis manuellement, proposer une fonction **ajouter_ami ()** qui admet en paramètres deux références, la première vers une Personne **pers** à qui on veut ajouter un ami **new_friend** (seconde référence constante). L'ajout est fait seulement si **new_friend** n'est pas déjà dans la liste d'amis de **pers**.

```
void ajouter_amis(Personne& pers, const Personne& new_friend) ;
```

Tester votre fonction en faisant afficher les amis avant et après l'appel de cette fonction pour une personne pers.

d) Malheureusement, il faut aussi proposer une fonction **enlever_ami ()** qui admet en paramètres deux références, la première vers une Personne **pers** à qui on veut enlever un ami **old_friend** (seconde référence constante). Le retrait est fait seulement si **old_friend** est dans la liste d'amis de **pers**.

```
void enlever_amis(Personne& pers, const Personne& old_friend);
```

La difficulté de cette question provient du fait que la personne à enlever n'est pas forcément la dernière de la liste d'amis. Cette difficulté peut être contournée simplement car il suffit d'échanger la place de **old_friend** avec le dernier élément de la liste d'amis avant d'utiliser la méthode **pop_back()** et le tour est joué.

Tester votre fonction en faisant afficher les amis avant et après l'appel de cette fonction pour une personne pers. Essayer différents scénarios : la personne est dans la liste d'amis ou pas, elle est en dernière position ou pas, etc…