Travaux Dirigés de C++.
Maitrise Informatique.

—*Julien Allali* - `allali@univ-mlv.fr` —

1 Exceptions.

▸ **Exercice 1.** *Déclarer une structure* `Cellule` *contenant un pointeur sur une chaîne de caractères, un entier ainsi qu'un pointeur sur la cellule suivante. La chaîne sera l'identifiant de la cellule et l'entier sa valeur.*

Écrire une fonction `createList` *qui prend en argument un tableau de chaînes de caractères. Pour chaque chaîne du tableau une cellule sera initialisée avec comme valeurs la chaîne (identifiant) et la position dans le tableau (valeur). Le tableau devra se terminer par un pointeur sur* `NULL`*. Tout argument invalide devra déclencher une exception. Tester cette fonction avec la liste d'arguments du programme.*

Écrire la fonction `cleanList` *qui prend en argument une liste chaînée et libère l'espace mémoire utilisé par celle-ci. L'argument de cette fonction sera une référence sur un pointeur de Cellule, il sera ré-initialisé à NULL en fin de fonction. Tester votre programme avec* `valgrind`*.*

Écrire une fonction `findList` *qui prend en argument une chaîne de caractères et une liste (par référence sur la première cellule). Cette fonction retourne une référence sur la cellule de la liste correspondant à l'argument. Tout argument invalide devra lever une exception. De même une recherche infructueuse lèvera une exception. Pour tester cette fonction vous lirez la valeur cherchée depuis l'entrée standard avec* `cin>>`*.*

Essayer d'utiliser la fonction `findList` *avec une référence constante comme tête de liste. Surcharger la fonction* `findList` *afin que l'on puisse utiliser cette fonction aussi avec une liste constante. On veillera à maintenir une cohérence au niveau des* `const`*.*

Écrire une fonction `concatList` *qui prend en argument deux listes et crée une* **nouvelle** *liste résultant de leur concaténation.*

2 Opérateurs.

▸ **Exercice 2.** *Opérateurs* `<<`

- *Écrire l'opérateur* `<<` *afin de pouvoir afficher une liste dans un flux de type* `ostream`*.*

3 Classe, constructeurs, destructeurs

▸ **Exercice 3.** *Point:*
Écrire une classe Point, contenant 2 valeurs x et y. Cette classe fera partie d'un espace de nom appelé "geometrie". La convention de nommage utilisée sera celle de Java.

▸ **Exercice 4.** *Constructeur, Destructeur : 20 minutes*
Écrire dans la classe Point, un constructeur sans argument et un destructeur dans lesquels on affichera un message sur la sortie standard. Puis écrire la fonction "main" suivante:

```
int main()
{
   Point p;
   return 1;
}
```

Constater le résultat.
Tester les cas suivants:

- *Modifier le constructeur pour qu'il prenne 2 arguments (x,y).*
- *Allouer dans le "main" un tableau de Point.*

- *Donner une valeur par défaut au argument du constructeur.*
- *Ecrire le constructeur par recopie. Puis la fonction suivante:*

  ```
  void f(Point p)
  {
  }
  ```

- *Typer le constructeur par recopie avec le mot clef "explicit".*
- *Trouver une solution.*

▸ **Exercice 5.** *explicit... : 15 minutes*
Ecrire dans un espace de nom "date", une classe Jour, Mois, Annee, (constructeur prenant un int en argument) puis une classe Date prenant en argument 3 objets : Jour, Mois, Annee.
Enfin écrire la fonction "main"

```
int main()
{
date::Date d(30,10,2001);
date::Date d2(2001,10,30);
return 0;
}
```

Proposer une utilisation d'explicit afin d'éviter la deuxième construction.

▸ **Exercice 6.** *Constructeur par recopie : 15 minutes*
Ecrire une classe Personne contenant les champs nom et prenom, un constructeur prenant en argument le nom et le prenom (allocation puis copie), et un destructeur désallouant la mémoire allouée pour le nom et le prénom. Puis écrire le "main" suivant:

```
void f(Personne p)
{}
void main()
{
Personne p(‘‘Lanfeust’’,’’Troy’’);
f(p);
}
```

Quel est le problème ?

▸ **Exercice 7.** *opérateurs: 30 minutes*
Ecrire l'operateur d'addition, soustraction pour la classe Point ainsi que l'operateur "<<".
Attention à l'utilisation des const et des références !

▸ **Exercice 8.** *Héritage, destructeur virtuel : 20 minutes*
Ecrire une classe A et une classe B avec leurs constructeurs et leurs destructeurs. La classe B héritant de A. Dans chaque constructeurs et destructeurs on affichera un message différent. Ecrire le "main" suivant puis tester. Typer les destructeurs avec "virtual", tester...

```
int main(void)
{
   A *a=new B();
   delete a;
   return 0;
}
```