Des indications sont données en fin de document.

1) Ecrire la classe **Chaine** représentant des chaines de caractères. Elle stocke les caractères dans un tableau de taille fixe (100 caractères max.), un zéro de fin de chaine marquant la fin des caractères utiles.

```
const int MAXCAR = 100;

class Chaine
{
public :
        Chaine ()                       { str[0] = '\0'; }          // chaine vide
        Chaine (const char* str2)       { strcpy (str, str2; }
        …
private :
        char  str  [MAXCAR + 1];
};
```

Les constructeurs sont donnés ci-dessus, il suffit de les recopier.

Ajouter en partie publique :

a) des opérateurs << et >> d'affichage et de saisie,
b) une fonction *int length() const* qui retourne la longueur de la chaine,
c) une fonction *char charAt (int i) const* qui retourne le caractère d'indice i,
d) une fonction *void setCharAt (int i, char newChar)* qui modifie le caractère d'indice i,
e) une fonction *void copyTo (char* dest) const* qui copie la chaine (str) dans le tableau dest,
f) une fonction *void copyFrom (const char* source)* qui copie source dans la chaine (str),
g) une fonction *bool equals (const Chaine& chaine2) const* qui teste l'égalité de la chaine et de chaine2,
h) une fonction *Chaine toLowerCase() const* qui retourne un objet Chaine qui contient les mêmes caractères que la chaine mais tout en minuscules.

Barème (5 points) : a(1), b(0,5), c(0,5), d(0,5), e(0,5), f(0,5), g(0,5), h(1).

2) Ecrire une classe **ChaineNoCasse** héritant de Chaine : une ChaineNoCasse est une Chaine insensible à la casse (majuscules/minuscules).

ChaineNoCasse n'ajoute pas de donnée par rapport à Chaine et elle fournit en partie publique :

a) un constructeur sans paramètre (chaine vide) et un constructeur recevant un const char*,
b) un opérateur == comparant deux ChaineNoCasse, qui sont déclarées égales si elles ont même séquence de caractères même si la casse est différente.

Barème (3 points) : a(1), b(2)

3) Ecrire la classe **Date**.

```
class Date
{
 public :
        …
private :
        int  jour, mois, annee;
};
```

avec :

a) un constructeur sans paramètre initialisant à 1/1/2000 et un constructeur recevant le jour, le mois et l'année,
b) des fonctions "get" et "set" pour le jour, le mois et l'année,
c) un opérateur << d'affichage affichant la date dans le format JJ/MM/AAAA, ex. 02/09/2011.

Barème (2 points) : a(0,5), b(0,5), c(1)

Ecrire la classe **Individu** :

```
class Individu
{
public :
…
private :
        Chaine nom;
        Date dateNaissance;
};
```

Ajouter en partie publique :

a) un constructeur sans paramètre initialisant le nom à "anonyme" et la date de naissance à 0/0/0 et un constructeur recevant le nom sous forme d'un const char* et la date de naissance sous forme de 3 entiers,
b) une fonction *void setIndividu (const char* leNom, int jourNaiss, int moisNaiss, int anneeNaiss)* qui modifie les données de l'individu,
c) une fonction *void afficher (bool miseEnForme) const* qui affiche le nom et la date de naissance; si miseEnForme est à false la fonction affiche le nom brut tel qu'il est stocké, si miseEnforme est à true elle affiche le nom avec une majuscule au début et le reste en minuscules; on donnera au paramètre miseEnForme une valeur par défaut de false.

Barème (4 points) : a(1), b(1), c(2)

4) Ecrire la classe **ChaineEtendue** qui hérite de Chaine sans ajouter de donnée mais en ajoutant des opérateurs et des fonctions.
En partie publique elle fournit :

a) un constructeur sans paramètre (chaine vide) et un constructeur recevant un const char*,
b) un opérateur ==,
c) un opérateur + de concaténation; il doit lancer une exception si la concaténation fait dépasser le nombre maximum de caractères, c'est-à-dire MAXCAR,
d) une fonction *int indexOf (char c) const* qui retourne l'indice de la première occurrence du caractère c, ou -1 si c n'est pas trouvé,
e) une fonction *ChaineEtendue substring (int begin, int end) const* qui retourne un objet ChaineEtendue qui est une sous-chaine de l'objet courant, begin et end étant les indices dans l'objet courant de début et fin de la sous-chaine; cette fonction doit lancer une exception si begin et end sont incorrects (hors des limites de la chaine, begin plus grand que end).

Barème(6 points) : a(1), b(1), c(2), d(1), e(2)

Indications :
On rappelle les fonctions *char tolower (char c)* et *char toupper (char c)* de <ctype.h> qui retournent respectivement la minuscule et la majuscule du caractère c passé en paramètre.