TD 8 : fichiers et images

Exercice 1.
On considère la fonction `mystere` suivante, avec un exemple d'appel :

```
void mystere(string zut) {
    ifstream bar(zut);
    string foo;
    int hop;
    while ( bar >> foo and bar >> hop ) {
        cout << foo << " " << 2 * hop << endl;
    }
    bar.close();
}
```

```
    mystere("fichier-test.txt");
```

(1) Deviner ce que cette fonction est censée faire. Préciser dans quel format doit être le fichier. Réécrire la fonction avec sa documentation en choisissant des noms informatifs pour elle-même et ses variables.

(2) Changer la fonction pour que chaque ligne de la sortie soit de la forme :

```
      Nom: Alfred, note sur 10: 7, note sur 20: 14
```

Exercice 2. (1) Implanter les deux fonctions suivantes.

```
/** calcule la moyenne des notes contenues dans un fichier
 *  Format: chaque ligne est de la forme "<nom> <note>"
 *  @param nomFichier le nom du fichier
 *  @return la moyenne des notes
 **/
float moyenne(string nomFichier);
```

```
/** lit les notes contenues dans un fichier et en fait un tableau
 *  Format: chaque ligne est de la forme "<nom> <note>"
 *  @param nomFichier le nom du fichier
 *  @return un tableau contenant les notes
 **/
vector<int> lit_notes(string nomFichier);
```

(2) Lorsque cela est possible, écrire un test.

(3) ♣ Comment pourrait-on tester la fonction du premier exercice ?

Exercice 3.
Pour manipuler informatiquement des images, on doit les numériser, c'est à dire transformer l'image analogique (continue) en une image numérique (discrète).

On distingue deux catégories de codage d'images. Premièrement, le codage vectoriel qui permet de coder l'image par un ensemble de formules mathématiques. Deuxièmement, le codage bitmap ou matriciel où l'image est représentée par une matrice finie de points (pixels) d'une certaine couleur.

Le fichier suivant contient une image noir et blanc au format PBM (*Portable Bit Map*).

Deviner comment fonctionne ce format de fichier et dessiner l'image correspondante.

```
P1
# CREATOR: GIMP PNM Filter Version 1.1
10 10
0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0
1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1
1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0
0 0 0 1 1 1 1 0 0 0
```

Indications : La première ligne précise le format du fichier (texte, image noir et blanc) ; la deuxième est un commentaire ; tout le reste a un rôle !

Exercice 4.
Le fichier suivant contient une image en couleur au format PPM (*Portable Pix Map*). Deviner comment fonctionne ce format de fichier et dessiner l'image correspondante.

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
3 2
255
0 255 0 255 255 255
255 0 0 0 255 0 255
255 255 255 0 0
```

Indication : Quelles sont les trois couleurs primaires usuelles ?

Exercice ♣ 5.
Implanter les fonctions suivantes :

```
/** compte le nombre de mots d'un fichier
 *  @param nomFichier le nom du fichier
 *  @return le nombre de mots contenus dans le fichier
 **/
int word_count(string nomFichier);
```

```
/** compte le nombre de lignes d'un fichier
 *  @param nomFichier le nom du fichier
 *  @return le nombre de lignes contenues dans le fichier
 **/
int line_count(string nomFichier);
```

```
/** affiche (sur la sortie standard) le contenu d'un fichier
 *  @param nomFichier le nom du fichier
 **/
void cat(string nomFichier);
```

```
/** copie le contenu d'un fichier dans un autre
 *  @param source le nom du fichier source
 *  @param destination le nom du fichier destination
 **/
void copy(string source, string destination);
```

Indication : la bibliothèque standard fournit la fonction suivante :

```
/** lit une ligne d'un flux et la stocke dans la chaîne de caractères s
 * @param f un flux entrant
 * @param s une chaîne de caractères
 * @return le flux entrant
 **/
istream getline(istream &f, string &s);
```

Exercice ♣ 6.
Deviner ce que fait la fonction `mystereEpais` suivante et écrire un test :

```
int mystereEpais(string rezut) {
    ifstream bla(rezut);
    int foo = 0;
    char y;
    while ( bla >> y ) {
        foo++;
    }
    bla.close();
    return foo;
}
```