Feuille de TD – III

Exercice 1 : Soit la fonction suivante :

```
int f(string t) {
   for (size_t i = 0; i < t.size(); i=i+1) {
      char c = t.at(i);
      if (c == ' ' or c == '\t' or c == '\n') continue;
      if (c >= '0' and c <= '9') return 1;
      if (c < 'a') return 2;
      else if (c <= 'z') return 3;
      return 2;
   }
   return 0;
}
```

1. Dans quel(s) cas la fonction peut-elle renvoyer `0` ? Décrire précisément et simplement ce que renvoie la fonction selon les différents cas.
2. Réécrire la fonction de manière à ce qu'elle ne comporte ni `continue`, ni `break` et qu'elle comporte un seul `return`, situé en fin de fonction.

Exercice 2 : On considère la gestion d'équipes de joueurs d'un sport collectif pour laquelle on mémorise différentes informations. Chaque **équipe** a un nom et on connaît son effectif (supposé limité à **25** joueurs). Pour un **joueur**, on connaît son nom, son numéro de licence (un entier strictement positif, unique sur l'ensemble des joueurs), un numéro qui désigne son équipe et son numéro de maillot (un entier compris entre 1 et 25). Au sein d'une équipe, chaque joueur est désigné par son numéro de licence. Si une équipe n'est pas au complet, les numéros de maillot ne sont pas forcément consécutifs.

On définit en C++ les types suivants :

```
typedef struct {
  string nom;       // nom du joueur
  size_t licence;   // numéro de licence  (un entier strictement positif)
  size_t maillot;   // numéro maillot : entre 1 et 25 (0 s'il n'est dans aucune équipe)
  size_t equipe;    // 1+l'indice de l'équipe dans le vecteur des équipes du championnat
                    // ou 0 s'il n'est dans aucune équipe.
} Joueur;

typedef struct {
  string nom;                 // le nom de l'equipe
  vector<size_t> joueurs;     // numéros de licence des joueurs de l'équipe
} Equipe;

typedef struct {
   vector<Joueur> joueurs;        // liste des joueurs
   vector<Equipe> equipes;        // liste des équipes
   size_t dernier;                // dernier numéro de licence actuellement attribué
} Championnat;
```

1. Écrire une fonction qui prend en entrée une référence sur un championnat et qui réinitialise ce championnat : en sortie, il ne contient plus aucune équipe et aucun joueur.

   `void initialiser(Championnat &c)`

2. Écrire une fonction qui prend en entrée un nom d'équipe et qui renvoie une valeur

correspondant à une nouvelle équipe avec aucun joueur.

```
Equipe creerEquipe(string nom)
```

3. Écrire une fonction qui prend en entrée une référence sur un championnat et un vecteur de noms d'équipe. Votre fonction ajouter au championnat des équipes avec ces noms, en vérifiant qu'on ne crée pas deux équipes avec le même nom (auquel cas on rejette la création des équipes dont le nom est déjà pris). La fonction renvoie le nombre d'équipes effectivement ajoutées.

```
size_t ajouter(Championnat &c, vector<string> lesNoms)
```

4. Écrire une fonction qui prend en entrée un championnat et un nom d'équipe et qui cherche s'il existe une équipe avec ce nom dans le championnat ; Elle renvoie alors le numéro de l'équipe dans le championnat (1+ son indice dans le vecteur des équipes), sinon `0`.

```
size_t numeroEquipe(Championnat const &c, string nomEquipe)
```

5. Écrire une fonction pour inscrire un **nouveau** joueur dans un championnat en l'inscrivant dans l'équipe dont on fournit le nom. On fournit aussi le numéro de maillot. Il ne doit pas déjà y avoir un joueur du championnat avec le même nom, l'équipe doit déjà exister et le numéro de maillot ne doit pas être déjà attribué dans cette équipe. On doit attribuer un numéro de licence disponible au joueur. La fonction renvoie `true` si on a pu inscrire le joueur et `false` sinon. En-tête :

```
bool inscrire(Championnat &c, string nomJ, string nomE, size_t num)
```

6. Écrire une fonction qui imprime les informations sur un joueur dont on fournit le numéro de licence. On imprime son nom , celui de son équipe et son numéro de maillot. On renvoie `true` s'il existe bien un tel joueur et `false` sinon.
   En-tête : `bool imprimeJoueur(const Championnat &c, size_t licence)`

7. Écrire une fonction qui imprime les informations sur tous les joueurs d'une équipe dont on passe le nom en paramètre. En-tête :

```
bool imprimeEquipe(const Championnat &c, string nom)
```

8. Écrire une fonction qui effectue un transfert de joueur (on fournit son numéro de licence) entre deux équipes en lui attribuant automatiquement **le plus petit numéro** de maillot disponible. La fonction renvoie `true` ou `false` selon que le transfert est possible ou pas. Si le transfert est possible, le joueur appartient maintenant à sa nouvelle équipe et il n'apparaît plus dans son ancienne équipe. En-tête est :

```
bool transfert(Championnat &c, size_t licence,
               string equipeA, string equipeB)
```