# NF16 - TP 4 – Les Arbres Binaires de Recherche

## Introduction

Dans ce TP, nous utiliserons les arbres binaires de recherche (ABR) et les listes simplement chaînées pour implémenter une gestion des étudiants inscrits à des UV.

L'ABR représente la liste des étudiant(e)s que l'on a inscrit à au moins une UV. Chaque nœud de l'arbre contient les informations permettant d'identifier un(e) étudiant(e), à savoir son nom et son prénom (on suppose qu'il n'y a pas deux étudiant(e)s ayant le même nom et prénom), ainsi que la liste des UV auquel il/elle est inscrit(e).

Le nom et le prénom constituent la clé du nœud, et l'ordre alphabétique sert de critère de comparaison entre deux nœud, par exemple :

*DUPOND MARCEL < DUPONT ALBERT < MARTIN JACQUES*

Une liste chaînée représente la liste des UV auxquelles un(e) étudiant(e) est inscrit(e). Chaque élément de la liste contient le code unique pour identifier l'UV (ex : NF16, AI01, etc…).

Important : les codes UV, les noms et les prénoms entrés par l'utilisateur seront convertis systématiquement en majuscules.

## A. Structures de données

Implémenter les structures de données et types suivants :

- La structure **`Element`** (et le type correspondant **`T_Element`**) qui comporte les champs :
  - **`code_uv`** de type **`char*`**
  - **`suivant`** de type **`struct Element*`**

- La structure **`Noeud`** (et le type correspondant **`T_Noeud`**) qui comporte les champs :
  - **`nom`** de type **`char*`**
  - **`prenom`** de type **`char*`**
  - **`listeInscriptions`** de type **`T_Element*`**
  - **`filsGauche`** de type **`struct Noeud`***
  - **`filsDroit`** de type **`struct Noeud`***

- Le type **`T_Arbre`**, qui représente l'ABR, de type **`T_Noeud`***

## B. Fonctions de base

1. Implémenter une fonction qui permet d'ajouter un élément dans une liste d'inscriptions à des UV, en veillant à **trier les éléments par ordre alphabétique du code UV** et garantir **l'unicité du code UV** dans une même liste. Cette fonction renvoie un pointeur vers le premier élément de la liste modifiée :

```
T_Element *ajouterInscription(T_Element *liste, char* code)
```

2. Implémenter une fonction qui permet d'inscrire un étudiant à une UV. Si l'étudiant n'est pas encore présent dans l'ABR, on devra créer le nœud qui le représente. Cette fonction renvoie un pointeur vers la racine de l'ABR :

```
T_Arbre inscrire(T_Arbre abr, char *nom, char *prenom, char *code)
```

.

3. Implémenter une fonction qui permet de charger dans l'ABR un fichier texte comportant une liste d'inscriptions : chaque ligne du fichier est sous la forme « NOM ;PRENOM ;CODE_UV ». Cette fonction renvoie un pointeur vers la racine de l'ABR :

```
T_Arbre chargerFichier(T_Arbre abr, char *filename)
```

4. Implémenter une fonction qui affiche la liste de tous les étudiants, classée par ordre alphabétique. Pour chaque étudiant on affichera également la liste de ses UV.

```
void afficherInscriptions(T_Arbre abr)
```

5. Implémenter une fonction qui affiche l'ensemble des étudiants inscrits à une UV :

```
void afficherInscriptionsUV(T_Arbre abr, char *code)
```

6. Implémenter une fonction qui permet de supprimer l'inscription d'un étudiant à une UV. Si l'étudiant n'a plus aucune inscription, on doit supprimer le nœud qui le représente de l'ABR. Cette fonction renvoie un pointeur vers la racine de l'ABR :

```
T_Arbre supprimerInscription(T_Arbre abr, char *nom, char *prenom, char *code)
```

## C. Programme Principal :

Programmer un menu qui propose les fonctionnalités suivantes :

1. **Inscrire un étudiant à une UV**
2. **Charger un fichier d'inscriptions**
3. **Afficher tous les étudiants**
4. **Afficher les inscrits à une UV**
5. **Supprimer une inscription**
6. **Quitter** (La mémoire allouée dynamiquement doit être libérée)

## Consignes générales :

### Sources

- À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés.
- L'organisation MINIMALE du projet est la suivante :
  - Fichier d'en-tête tp4.h, contenant la déclaration des structures/fonctions de base,
  - Fichier source tp4.c, contenant la définition de chaque fonction,
  - Fichier source main.c, contenant le programme principal.

### Rapport

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.
- Un exposé succinct de la complexité de chacune des fonctions implémentées.
- Votre rapport et vos fichiers source feront l'objet d'une remise sur Moodle dans l'espace qui sera ouvert à cet effet quelques jours suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).