STRUCTURES ET LISTE CHAÎNÉE

Dans ce TP nous allons créer une mini-application de gestion d'un carnet d'adresses.

Chaque entrée du carnet d'adresses comportera les informations suivantes :

— Nom
— Prénom
— Sexe (M ou F)
— Numéro de téléphone fixe
— Numéro de téléphone portable
— Un pointeur sur le conjoint

On représentera le carnet d'adresses par une ***liste chaînée*** de contacts. Le dernier champ de cette structure sera donc appelé `suivant` et sera un pointeur vers la cellule suivante.

1. Proposer une structure `contact` pour représenter un contact.
   A l'aide d'un `typedef`, créer pour cette structure l'*alias* `CELLULE`.
2. Spécifier et programmer une fonction qui permet d'afficher les informations relatives à un contact.
   Prototype : `void afficheContact(CELLULE * c);`
3. Créer deux contacts (par exemple « Homer Simpson », numéros à choisir, et « Marge Simpson » dont le conjoint est « Homer Simpson ») à l'aide d'une fonction `new` dont le prototype peut-être

   ```
   CELLULE * new(char *n, char * pn, char * f, char * pt,  int s, CELLULE * c);.
   ```

   Les arguments peuvent varier selon les types des champs choisis dans votre structure.
   Le pointeur vers le `conjoint` sera `c` et celui vers le `suivant` sera initialisé à `NULL`.
   Mettre les noms en majuscule (à l'aide d'une fonction utilisant `toupper` de `ctype.h` par exemple) peut être une bonne idée... Tester alors votre fonction d'affichage.
4. Spécifier et programmer une fonction qui permet d'ajouter un contact à un carnet d'adresses, maintenu dans l'ordre alphabétique. Prototype :

   ```
   CELLULE * rajoutOrdre(CELLULE * tete, CELLULE * cell);
   ```

   où on aura d'abord créé la liste en créant le pointeur de tête `CELLULE * tete=NULL;` au début du `main`.
   Cette fonction renvoie l'adresse de la nouvelle « tête ».
5. Définir une fonction récursive `liberer_liste` permettant de libérer la mémoire allouée pour chaque élément d'une liste. La fonction prend en paramètre l'adresse de la tête de la liste à libérer.
6. Spécifier et programmer une fonction

   ```
   int est_dans_liste(CELLULE * tete, char * nom,char * prenom);
   ```

   « booléenne » qui renvoit `1` si la personne est déjà dans le répertoire et `0` sinon.
7. Spécifier et programmer une fonction qui permet de créer un contact en demandant à l'utilisateur les informations nécessaires (excepté le conjoint), *s'il n'est pas encore dans la liste.*
8. Spécifier et programmer une fonction qui permet de supprimer un contact dont l'utilisateur a donné le nom et le prénom (la première occurrence), *s'il est bien dans la liste.*
9. Spécifier une fonction qui permet de « mettre en couple » deux contacts *différents* du carnet d'adresse dont on donne le nom et le prénom (la première occurrence), *s'ils sont bien dans la liste.*
10. Créer un menu qui propose les choix suivants :

```
MENU PRINCIPAL

1 - Saisir un contact
2 - Retirer un contact
3 - Lister les contacts
4 - Mettre en couple deux contacts
5 - Quitter

Votre choix >
```