NF16 - TP3 : Les listes linéaires chaînées

Enoncé:

Le département de l'Oise souhaite mettre en place un système pour gérer son stock de vaccins contre la Covid-19 en temps réel.
Pour ce faire, on va enregistrer pour chaque commune possédant un centre de vaccination, le nombre de vaccins dont la commune dispose.
En outre, le nombre de vaccins disponibles pour une commune sera réparti sur plusieurs semaines : cela correspond aux prévisions de vaccins à administrer chaque semaine.

Ces données seront enregistrées comme suit :

- un tableau dont les éléments représentent les marques de vaccins disponibles dans le département
- chaque élément du tableau indique la marque du vaccin, ainsi qu'une liste chaînée qui représente la liste des communes possédant un stock de ce vaccin
- un élément d'une liste chaînée des communes va indiquer le nom de la commune, ainsi qu'une liste chaînée qui représente la répartition du stock des vaccins de la commune pour chaque semaine de vaccination planifiée
- un élément d'une liste chaînée des semaines va indiquer le numéro de semaine ainsi que le nombre de vaccins planifiés pour cette semaine

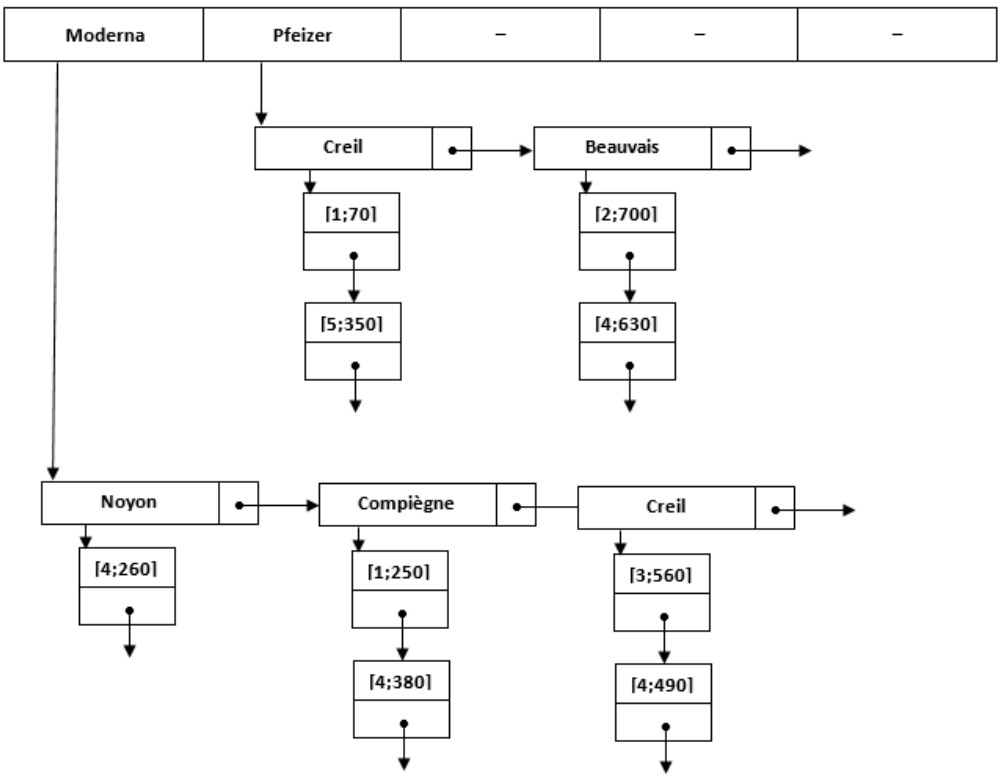
Exemple :

Pour le vaccin Moderna, le département dispose des stocks suivants :

- Compiègne : 250 vaccins en semaine 1, 380 vaccins en semaine 4
- Creil : 560 vaccins en semaine 3, 490 vaccins en semaine 4
- Noyon : 260 vaccins en semaine 4

Pour le vaccin Pfeizer, le département dispose des stocks suivants :

- Beauvais : 700 vaccins en semaine 2, 630 vaccins en semaine 4
- Creil : 70 vaccins en semaine 1, 350 vaccins en semaine 5



A. Structures de données :

Définir les structures de données suivantes :

- La structure **`semaine_elt`**, et le type associé **`t_semaine_elt`,** qui comporte les champs :
  - **`numero_semaine`** de type *unsigned int* (compris entre 1 et 53)
  - **`nombre_vaccins`** de type *unsigned int* (strictement positif !)
  - **`suivant`** de type *struct semaine_elt**

- La structure **`ville_elt`**, et le type associé **`t_ville_elt`,** qui comporte les champs :
  - **`nom_ville`** de type *char**
  - **`nombre_vaccins_total`** de type *unsigned int* (strictement positif !)
  - **`semaines_planifiees`** de type *t_semaine_elt**
  - **`suivant`** de type *struct ville_elt**

- La structure **`vaccin_elt`**, et le type associé **`t_vaccin_elt`,** qui comporte les champs :
  - **`marque`** de type *char**
  - **`villes_dispo`** de type *t_ville_elt**

B. Fonctions à implémenter :

1. Ecrire une fonction qui permet de créer un nouvel élément semaine, à partir d'un numéro de semaine et un nombre de vaccins, et renvoie un pointeur vers la nouvelle structure :

```
t_semaine_elt *creerSemaine(int num_semaine, int nb_vaccins);
```

2. Ecrire une fonction qui permet de créer un nouvel élément ville, à partir d'un nom de ville, et renvoie un pointeur vers la nouvelle structure :

```
t_ville_elt *creerVille(char *ville);
```

3. Ecrire une fonction qui permet de créer un nouvel élément vaccin, à partir d'une marque de vaccin, et renvoie un pointeur vers la nouvelle structure :

```
t_vaccin_elt *creerVaccin(char *marque);
```

4. Ecrire une fonction qui permet d'ajouter un nombre de vaccins pour une semaine donnée, et renvoie un pointeur vers le premier élément de la liste modifiée :

```
t_semaine_elt *ajouterVaccinS(t_semaine_elt *liste, int semaine, int nb_vaccins);
```

Si la semaine est déjà dans la liste, on additionne le nouveau nombre et le nombre de vaccins déjà planifiés. **L'ajout se fait en respectant le tri par ordre croissant de numéros de semaine !**

5. Ecrire une fonction qui permet de déduire un nombre de vaccins pour une semaine donnée, et renvoie un pointeur vers le premier élément de la liste modifiée :

```
t_semaine_elt *deduireVaccinS(t_semaine_elt *liste, int semaine, int nb_vaccins);
```

Pensez à gérer le cas où l'on retire tous les vaccins pour la semaine indiquée !

6. Ecrire une fonction qui permet d'ajouter un nombre de vaccins pour une ville et une semaine données, et renvoie un pointeur vers le premier élément de la liste modifiée :

```
t_ville_elt *ajouterVaccinV(t_ville_elt *liste, char* ville, int semaine, int nb_vaccins);
```

L'ajout dans la liste des villes se fait en respectant le tri par ordre croissant du total de vaccins !

7. Ecrire une fonction qui permet de déduire un nombre de vaccins pour une ville et une semaine données, et renvoie un pointeur vers le premier élément de la liste modifiée :

```
t_ville_elt *deduireVaccinV(t_ville_elt *liste, char* ville, int semaine, int nb_vaccins);
```

Pensez à gérer le cas où il n'y a plus de vaccin disponible pour la ville indiquée !

8. Ecrire une fonction qui permet d'afficher tous les stocks disponibles et leur planification pour une marque donnée :

```
void afficherStock(t_vaccin_elt *vaccin);
```

L'affichage se fera sous la forme :

```
Moderna :
 --- Noyon [Total = 260]
        --- semaine 4 : 260
 --- Compiègne [Total = 630]
        --- semaine 1 : 250
        --- semaine 4 : 380
 --- Creil [Total = 1050]
        --- semaine 3 : 560
        --- semaine 4 : 490
```

9. Ecrire une fonction qui permet d'afficher la planification pour une semaine et un vaccin donnés :

```
void afficherPlanification(t_vaccin_elt *vaccin, int semaine);
```

L'affichage se fera sous la forme :

```
Moderna :
 --- semaine 4
        --- Noyon : 260
        --- Compiègne : 380
        --- Creil : 490
```

10. Ecrire une fonction qui permet de fusionner les stocks d'un vaccin A et un vaccin B. On ne doit pas modifier les stocks existants, mais le résultat de la fusion sera stocké dans une nouvelle entrée ayant une marque de vaccin fictive « vaccinA_vaccinB ». La fonction renvoie un pointeur vers le premier élément de la liste créée :

```
t_vaccin_elt *fusionnerStocks(t_vaccin_elt *vaccinA, t_vaccin_elt *vaccinB);
```

Vous attacherez une importance particulière à optimiser l'algorithme utilisé : inspirez-vous des exemples vus en cours et en TD !

C. Interface:

Utiliser les fonctions précédentes pour écrire un programme permettant de manipuler les matrices creuses. Le programme propose un menu qui contient les fonctionnalités suivantes :

1. Initialiser la liste des marques de vaccin disponibles
2. Ajouter un nombre de vaccins dans le stock d'une marque
3. Retirer un nombre de vaccins du stock
4. Afficher le stock d'un vaccin
5. Afficher la planification pour une semaine
6. Fusionner les stocks de deux marques de vaccins
7. Quitter

Important : votre programme principal devra gérer un **tableau de 10 éléments de type *t_vaccin_elt** ** afin de gérer les différentes marques de vaccin proposées

Vous devrez également ajouter des contrôles dans vos différentes fonctionnalités pour vérifier que l'utilisateur saisit des valeurs cohérentes, et apporter une assistance à la saisie quand c'est possible.

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 tp3.h, contenant la déclaration des structures/fonctions de base,
- Fichier source tp3.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 trois fichiers feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet (un seul rendu de devoir par binôme).