Etape 4 : Tri d'une liste de « struct »

Objectif : Créer une liste de « T_personne » et la trier sur différents critères, croissants ou décroissants.

Dans le support de démonstration sur les fonctions de pointeurs, vous y avez trouvé le type `T_personne` dans *types.h* :

```
typedef struct {
    int id;
    char nom[20];
    int taille;   //en cm
    float masse;  //en Kg
    int annee,mois,jour;
} T_personne;
```

Et des fonctions de comparaisons entre champs :

```
bool comp_taille_AsupB ( T_personne a, T_personne b);
bool comp_taille_AinfB ( T_personne a, T_personne b);
bool comp_taille_AegaleB ( T_personne a, T_personne b);

bool comp_masse_AsupB  ( T_personne a, T_personne b);
bool comp_masse_AinfB  ( T_personne a, T_personne b);
bool comp_masse_AegaleB  ( T_personne a, T_personne b);
```

Actions à faire :

1. Repartez de votre code de l'étape 3 (tri d'une liste sans déplacer les données) ou de sa correction, ajoutez-y les fichiers types.h/types.c et utils.h/utils.c (sous codebloc : menu project | add file).

2. Dans listeDouble.h, modifiez le type du champ `pdata` du type `T_cellule` pour en faire un pointeur sur `T_personne`. Comme vous aurez donc besoin de connaitre le type `T_personne` dans listeDouble.h, ajoutez un `#include "types.h"` dans celui-ci.

3. Mettez `getOccurences` et `tri_selection_liste` en commentaires. Puis mettez à jour les entêtes de listeDouble.h et le code (listeDouble.c) des autres fonctions qui utilisaient le champ `pdata`, puisque celui-ci est devenu un pointeur sur `T_personne`.

4. Afin de disposer facilement d'une `T_liste` de `T_personne`, écrivez dans listeDouble.h/.c la fonction :

   `T_liste creerListeNElem(int taille)`

   qui pourra utiliser elle-même la fonction `getPersAlea` de *type.h* pour créer des « personnes » fictives.

5. La fonction `getOccurences` a droit à un traitement particulier : nous allons rechercher les occurrences sur un seul champ. Pour ce faire ajoutez une fonction de comparaison (d'égalité) en paramètre. Vous pouvez (re)regarder le support « prérequis étape 4 : pointeurs de fonctions » et le code de démonstration.

Testez votre nouvelle version de `getOccurences`.

6. La fonction de tri de l'étape 3 a droit à la même démarche. Ajoutez une fonction de comparaison en paramètre puis testez-la avec `comp_taille_AsupB` puis `comp_masse_AinfB`.

   Il est possible que votre tri soit décroissant au lieu d'être croissant (ou l'inverse) : vérifier l'ordre des paramètres lors de vos appels à `comp_masse_AinfB` et `comp_taille_AsupB`, afin de respecter le sens de la comparaison initiale dans le code du tri.

7. Regardez la correction, même si vous y êtes arrivé seul.