SÉRIE N° 3 : LES STRUCTURES -SUITE-

EXERCICE 1: Des wagons dans des trains ;
Dans cet exercice, nous allons considérer deux structures imbriquées :

Soit une structure nommée « ***wagon*** », contenant deux entiers positifs, « ***s*** » et «***p*** » qui indiquent respectivement le nombre de sièges et le nombre de passagers dans le wagon.

1. Définir la structure « ***wagon*** », qui aura comme type synonyme « ***WAGON*** ».
2. Ecrire une fonction ***int wagon_places_dispo(WAGON W)*** qui retourne le nombre de places disponibles dans un wagon W.
3. Ecrire une fonction ***int wagon_ajouter_passagers(WAGON *W, int x)*** qui permet d'ajouter des passagers (dont le nombre correspond à l'entier positif x fourni en second argument d'entrée) dans un wagon (dont on connaît l'adresse, fournie en premier argument) et qui renvoie :
   - ✓ 0 si tous les passagers ont trouvé une place,
   - ✓ le nombre de passagers qui n'ont pas trouvé de places.
4. Ecrire une fonction ***void wagon_init(WAGON T[], int n)*** qui permet d'initialiser un tableau T de ***WAGON*** de taille ***n*** : le nombre de sièges par défaut sera de 90 places et le nombre de passagers, 0.

Soit la structure nommée « ***train*** » composée de deux champs : le premier sera un entier positif indiquant le nombre de wagons, et le second, un tableau T de type WAGON (taille max :50).

5. Définir la structure « ***train*** » de type synonyme « ***TRAIN*** ».
6. Ecrire une fonction ***void train_init(TRAIN *Tr)*** qui permet d'initialiser le train (on remplira les champs du train : le nombre de sièges par défaut sera de 90 places et le nombre de passagers, 0).
7. Ecrire une fonction ***TRAIN train_ajouter_train(TRAIN Tr1, TRAIN Tr2)*** qui permet de fusionner ensemble deux trains. Le résultat de cette somme est un nouveau train contenant tous les wagons des 2 autres.
8. Ecrire une procédure ***void train_ajouter_wagon(TRAIN *Tr, WAGON W)*** qui permet d'ajouter un wagon à un train. Le train résultant sera égal au train précédent avec un wagon supplémentaire (correspondant au wagon fourni en argument d'entrée).
9. Ecrire une procédure ***void train_ajouter_passagers(TRAIN *Tr, int x)*** qui permet d'ajouter des passagers (dont le nombre correspond à l'entier positif x fourni en second argument d'entrée) au train (dont on connaît l'adresse via le premier argument) en commençant par remplir le premier wagon, puis le second et ainsi de suite tant qu'il reste des passagers à placer ; s'il n'y a plus de places disponibles dans le train, on ajoute un nouveau wagon contenant le nombre de passagers restant à placer.
10. Ecrire une procédure ***void train_tri_wagon(TRAIN *T)*** qui trie les wagons du train selon l'ordre croissant du nombre de passagers avec la méthode de votre choix.