APP Récursivité

Exercice 1 :

1) Écrire une fonction récursive **inverserRec( )** qui permet d'inverser un tableau d'entiers
2) Écrire une fonction récursive **palindromeRec( )** qui permet de vérifier si un mot est palindrome ou non.

   Un mot est palindrome s'il se lit de la même manière de gauche à droite et de droite à gauche. Exemples : elle, non, kayak, radar, solos, …
3) Écrire une fonction récursive **pgcdRec( )** qui permet de calculer le PGCD de deux entiers positifs selon l'algorithme d'Euclide :

   PGCD(a, b) = PGCD(b, r) où *r* est le reste de la division euclidienne de *a* par *b*,

   sachant que *PGCD(a,0) = a*

Exercice 2 :

On considère une liste simplement chaînée d'étudiants où chaque étudiant est caractérisé par son nom et sa moyenne

1. Écrire une fonction récursive qui permet d'afficher les étudiants de la liste en commençant par l'étudiant se trouvant à la tête de la liste.

   void afficher_liste (liste l)
2. Écrire une fonction récursive qui permet d'afficher les étudiants de la liste en commençant par l'étudiant se trouvant à la fin de la liste.

   void afficher_liste_inversee (liste l)
3. Écrire une fonction récursive qui permet d'ajouter un étudiant en fin de liste

   liste ajouter_etudiant (liste l, Etudiant E)
4. Écrire une fonction récursive qui permet de supprimer l'étudiant dont le nom est passé en paramètre, de la liste d'étudiants.

   liste supprimer_etudiant (liste l, char *nom)

Exercice 3 : Tri par fusion (optionnel)

Le tri par fusion est un algorithme de tri qui se base sur l'approche de résolution de problèmes "diviser pour régner".

Le principe du tri par fusion appliqué sur un tableau de *n* éléments est le suivant :

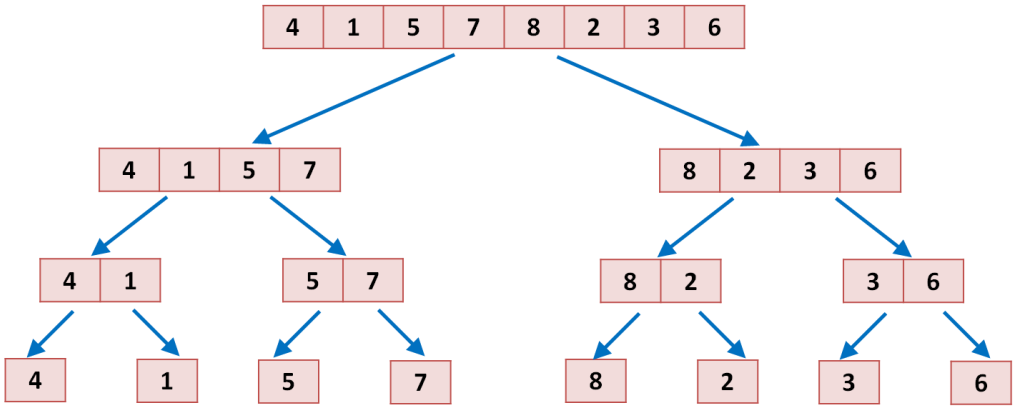
- **Diviser** : diviser le tableau de n éléments à trier en deux tableaux de n/2 éléments
- **Régner** : trier les deux tableaux obtenus de façon récursive

  La récursivité s'arrête à un tableau contenant un seul élément, donc déjà trié
- **Combiner** : fusionner les deux tableaux obtenus pour obtenir la réponse triée (il faut écrire une fonction itérative dédiée à cette opération), d'où le nom de tri fusion.

- Implémenter la fonction récursive **TriFusion(int Tab[], int n, int deb, int fin)** qui, étant donné un tableau *Tab* de *n* entiers, permet de trier les éléments du sous tableau commençant de l'indice deb jusqu'à l'indice fin.
- Implémenter la fonction itérative **Interclasser(int Tab[], int n, int deb, int milieu, int fin)** qui, étant donné un tableau *Tab* de *n* entiers, permet d'interclasser les deux sous-tableaux commençant respectivement de l'indice *deb* jusqu'à *milieu* et de l'indice *(milieu +1)* jusqu'à *fin*.

Illustration du fonctionnement sur un exemple :

Soit le tableau *Tab* suivant. Les récursions découpent notre tableau jusqu'à l'obtention de tableaux unaires :



Puis, à la remontée des récursions, les tableaux sont fusionnés :