Travaux Pratiques
Feuille de TP N° 2

Objectifs d'apprentissage :

- **Maîtriser la manipulation des listes simplement chaînées**

On donne les définitions suivantes pour représenter des listes chaînées d'entiers :

```
typedef struct cellule
{
  int contenu;
  struct cellule *suivant;
} cellule;
typedef cellule *Liste;
```

1. Écrire un programme principal qui lit la valeur d'un entier n ≥ 1, puis crée la liste des nombres impairs de 2n-1 à 1 : (2n-1, …, 3, 1) par insertion au début. Afficher ensuite cette liste. Par exemple si n = 4, on créera la liste (7, 5, 3, 1) et l'on affichera sous le format : `7->5->3->1->NULL`.
2. Écrire une fonction « impairs » qui crée la liste (2n-1, …, 3, 1) par insertion des éléments au début. La fonction devra uniquement retourner la liste sans l'afficher.
3. Écrire une procédure itérative « afficher » qui affiche les éléments d'une liste sous le format indiqué dans la question 1.
4. Écrire une version récursive de la procédure « afficher ».
5. Refaire la question 1) en utilisant les fonctions précédentes.
6. Écrire un programme principal qui lit la valeur d'un entier n ≥ 0, puis crée la liste des nombres pairs de 0 à 2n : (0, 2, …, 2n) par insertion à la fin. Afficher ensuite cette liste. Par exemple pour n = 3, on créera la liste (0, 2, 4, 6) et l'on affichera sous le format : `0->2->4->6->NULL`.
7. Écrire une fonction « pairs » qui crée la liste (0, 2, …, 2n) par insertion à la fin (sans l'afficher).
8. Écrire un programme principal qui effectue les tâches suivantes :
   a. Lit la valeur d'un entier $n \geq 0$.
   b. Crée la liste (0, 2, …, 2n) en appelant la fonction « pairs ».
   c. Supprime de cette liste tous les multiples de 3.
   d. Affiche la liste obtenue.
9. Écrire une fonction itérative « somme » qui calcule la somme des éléments d'une liste L.

10. Écrire une version récursive de la fonction « somme ».
11. On suppose que L est une liste triée dans l'ordre croissant, mais elle contient des éléments dupliqués. Écrire une fonction itérative « supprimerDup » qui supprime de L tous les éléments dupliqués. Par exemple, si L = (1, 1, 2, 2, 3, 5, 5, 5, 8, 8, 9), alors après l'appel de la fonction « supprimerDup », la liste L deviendra égale à (1, 2, 3, 5, 8, 9).
12. Écrire une version récursive de la fonction « supprimerDup ».
13. Écrire une fonction itérative « appartient » qui teste si un élément e appartient ou non à une liste L.
14. Écrire une version récursive de la fonction « appartient ».
15. Écrire une fonction « croiser » qui prends deux liste $L_1 = (e_1, \ldots, e_N)$ et $L_2 = (a_1, \ldots, a_N)$ ayant la même longueur N et construit la liste $L = (e_1, a_1, e_2, a_2, \ldots, e_N, a_N)$. Si les deux listes n'ont pas la même longueur, la fonction retourne la liste $(e_1, a_1, e_2, a_2, \ldots, e_K, a_K)$ où K est la plus petite longueur et la complète par les éléments de la liste ayant la plus grande longueur. Par exemple, croiser((1, 7, 3, 5), (2, 0)) donne (1, 2, 7, 0, 3, 5).