

TP 2 :**Les listes****Objectifs**

Manipuler des variables de type Liste chaînée.

Rappel :

L'allocation dynamique se fait par le biais de la fonction `malloc()` qui permet d'allouer un certain nombre d'octets et qui renvoie un pointeur sur cette case mémoire allouée.

Voici le prototype de la fonction `malloc()` :

```
void *malloc(size_t size);
```

Exercice 1 :

1. Ecrire la fonction saisir permettant de remplir un tableau T de N entiers.
2. Ecrire une fonction Copier qui permet de copier un tableau de N entiers dans une liste simplement chaînée
3. Ecrire une fonction Paire permettant de stocker les nombres pairs du tableau T dans une liste doublement chaînée
4. Ecrire un programme principal permettant de tester ces fonctions

Exercice 2 :

On définit une matrice creuse comme étant une matrice dont plus que la moitié des éléments sont nuls.

0	7	0	5	0	0
6	0	0	0	0	5
0	0	1	0	0	0
0	0	0	0	0	0

Pour minimiser l'espace occupé par ce type de matrice, on choisit de les représenter sous forme d'un tableau de listes chaînées, de sorte que la $i^{\text{ème}}$ liste chaînée contient les éléments non nuls

de la ligne i de la matrice et chacun d'eux accompagné du numéro de la colonne où il se trouve.

$T[0]$	<div>7 1</div>	<div>5 3</div>
$T[1]$	<div>6 0</div>	<div>5 5</div>
$T[2]$	<div>1 2</div>	
$T[3]$		

Tâches à faire :

1. Définir la structure qui sera utilisée pour les cellules des listes chaînées;
2. Transformer une matrice M de taille $n \times m$ en tableau de listes chaînées T comme défini précédemment;
3. Afficher un élément $M[i][j]$ à partir de T .

Exercice 3 :

- Ecrire la fonction : longueur_totale qui calcule la longueur d'une liste doublement chaînée.
- Ecrire les fonctions : longueur_apres et longueur_avant qui, pour une liste L et un pointeur P sur un élément de la liste, calculent le nombre d'éléments après et avant la liste P .