REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Devoir Surveillé ■ Examen □		Session principale ■	
		Session de contrôle □	
Matière :	Algorithmique et structures de données II	**Semestre :** 2	
Enseignant(s) :	Majdi Jribi et Rabaa Youssef	**Date:** 09 Mars 2022	
Filière(s) :	MPI	**Durée:** 1h30	
Barème :	05-15	**Documents :** autorisés □	
Nombre de pages :	02	non autorisés ■	

Exercice 1 (05 pts)

Un nombre romain peut se composer des chiffres suivants(représentés par des lettres **majuscules**):

Chiffre romain	M	D	C	L	X	V	I
Valeur	1000	500	100	50	10	5	1

Un nombre romain est interprété grâce aux règles suivantes:

- La valeur du nombre est égale à celle du chiffre romain si celui-ci est composé d'un seul chiffre.
- Si le nombre contient au moins deux chiffres, on considère les deux chiffres les plus à gauche :
  - Si le 1er chiffre (le plus à gauche) est supérieur ou égal au 2ème, alors la valeur du nombre romain correspond à la **somme** de la valeur du 1er chiffre et de la valeur du nombre composé de tous les chiffres restants du nombre.
  - Si le 1er chiffre est inférieur au 2ème, alors la valeur du nombre correspond à la **différence** entre la valeur du nombre composé des chiffres restants et la valeur du 1erchiffre (le plus à gauche).

Exemples:

- **IIII = 4 ; IV = 4**
- **VIIII =9 ; IX = 9**
- **MIM= 1999 ; MCMXCIX = 1999**

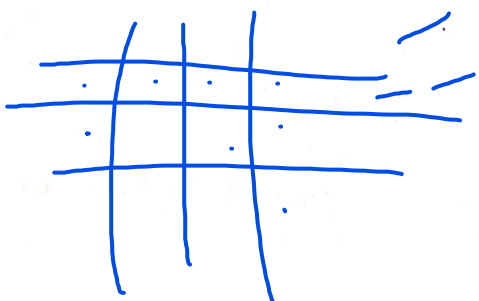
1. Ecrire en **langage C** une **fonction itérative**
   `int evalue_chiffre_romain (Char c)`
   qui renvoie la valeur numérique associée au chiffre romain passé en paramètre sous la forme d'un simple caractère.
2. Ecrire en **langage C** une **fonction récursive**
   `int evalue_nombre_romain(char * ch)`
   qui reçoit en paramètre un nombre romain représenté par une chaîne de caractères, et qui renvoie le résultat de l'évaluation de ce nombre.

Remarque:on suppose que la syntaxe du nombre romain donné est correcte.

Exercice 2 (15 pts)

Nous souhaitons modéliser les matrices à l'aide des listes chainées. Pour cela nous définissons les structures de données suivantes :

```
typedef struct Noeud {
      float valeur;
      struct Noeud *next_col;
      struct Noeud *next_lig;
} Noeud;
```



```
typedef struct {
    Noeud *first;
    int nb_lignes;
    int nb_colonnes;
} Matrice;
```

La structure **Noeud** contient trois champs :

- Un champs de données **valeur** de type réel.
- Un champ pointeur sur la colonne suivante **next_col** qui est un pointeur sur le nœud de la colonne suivante au nœud actuel.
- Un champ pointeur sur la ligne suivante **next_lig** qui est un pointeur sur le nœud de la ligne suivante au nœud actuel.

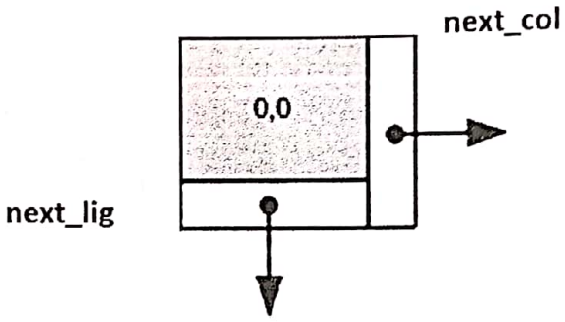


Figure 1 : Illsutration de la structure Noeud

Une matrice (qui correspond au type **Matrice**) est représentée par une structure de données contenant trois champs :

- Un champ **first** qui est pointeur sur Noeud et qui correspond à l'adresse de la matrice. C'est l'adresse de l'élément le plus à gauche et en haut de la matrice **(comme mentionné par la figure 2).**
- Un champ **nb_lignes** de type entier qui indique le nombre de lignes de la matrice.
- Un champ **nb_colonnes** de type entier qui indique le nombre de colonnes de la matrice.

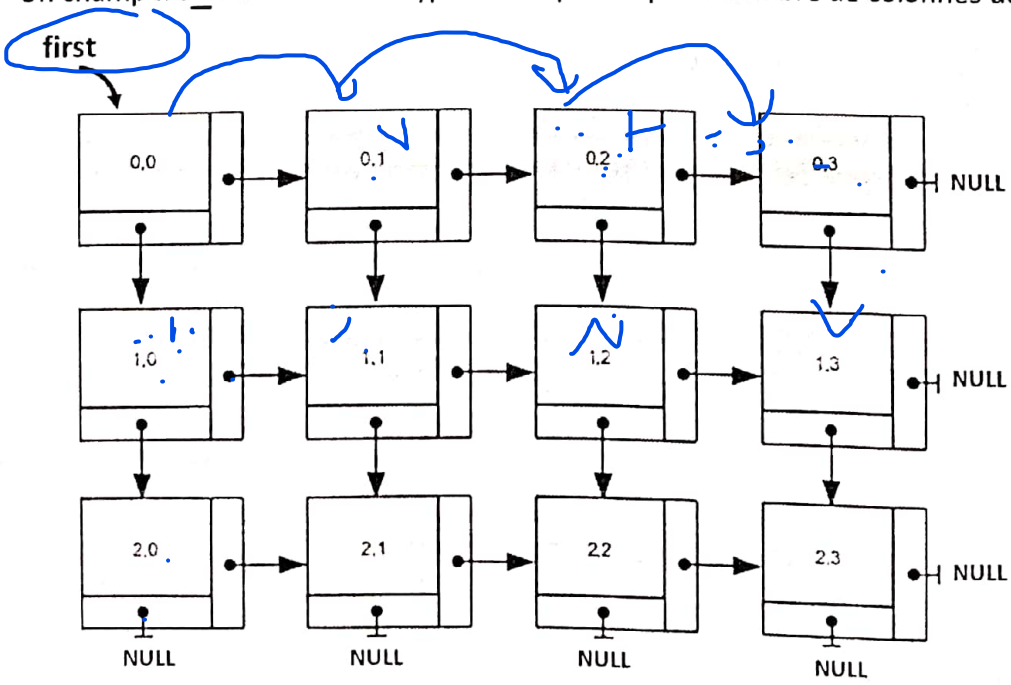


Figure 2 : Illustration d'une matrice avec nb_lignes=3 et nb_colonnes=4

1- Ecrire en **langage C** la fonction
`Matrice Creation (Matrice M, int nb_lignes, int nb_colonnes)`
qui permet de créer une Matrice **M** de **`nb_lignes`** lignes et de **`nb_colonnes`** colonnes. Les champs de données seront saisies par l'utilsateur.

2- Ecrire en **langage C** la fonction
`Matrice Inserer_Lig (Matrice M, int nb_lignes, int nb_colonnes, Matrice Ligne, int pos)`
qui permet d'inserer dans la Matrice **M** de **`nb_lignes`** lignes et de **`nb_colonnes`** colonnes une matrice **`Ligne`** (contenant 1 ligne et **`nb_colonnes`** colonnes) dans une ligne d'indice **`pos`** dans la matrice **M**)

3- Ecrire en **langage C** la fonction
`Matrice Inserer_Col (Matrice M, int nb_lignes, int nb_colonnes, Matrice Colonne, int pos)`
qui permet d'inserer dans la Matrice **M** de **`nb_lignes`** lignes et de **`nb_colonnes`** colonnes une matrice **`Colonne`** (contenant **`nb_lignes`** lignes et 1 colonne) dans une colonne d'indice **`pos`** dans la matrice **M**)

4- Ecrire en **langage C** la fonction
`Matrice Supp_Lig (Matrice M, int nb_lignes, int nb_colonnes, int pos)`
qui permet de supprimer de la Matrice **M** de **`nb_lignes`** lignes et de **`nb_colonnes`** colonnes la ligne (contenant 1 ligne et **`nb_colonnes`** colonnes) d'indice **`pos`** dans la Matrice **M**.

5- Ecrire en **langage C** la fonction
`Matrice Supp_Col (Matrice M, int nb_lignes, int nb_colonnes, int pos)`
qui permet de supprimer de la Matrice **M** de **`nb_lignes`** lignes et de **`nb_colonnes`** colonnes la colonne (contenant **`nb_lignes`** lignes et 1 colonne) d'indice **`pos`** dans la Matrice **M**.