Travaux dirigés
Programmation en C
—IMAC Première Année—

Cette feuille constitue l'annexe de la feuille de rattapage.

▸ **Exercice 1.** ***Compter les malloc() et les free()***

En remplaçant dans les programmes de la feuille précédente les appels à **malloc()** *et à* **free()** *par les fonctions ci-dessous, vérifiez que, à la fin du programme, vous avez effectué autant de* **malloc()** *que de* **free()**.

```
int global_allouee=0;
void *mon_malloc(size_t taille)
{
  global_allouee++;
  return malloc(taille);
}
void mon_free(void *ptr)
{
  global_allouee--;
  free(ptr);
}
```

▸ **Exercice 2. Factorielle**

La factorielle *d'un entier positif ou nul est définie par :*

$$n! = \begin{cases} 1 & \text{si} \quad n = 0 \\ (n-1)! * n & \text{si} \quad n > 0 \end{cases}$$

1. *Écrire la fonction itérative* **unsigned int factorielle_i(unsigned int n)** *qui renvoie la factorielle d'un entier positif.*
2. *Écrire la fonction récursive* **unsigned int factorielle_r(unsigned int n)** *qui renvoie la factorielle d'un entier positif.*

▸ **Exercice 3.** ***Nombres de Fibonacci***
Les nombres de Fibonacci sont définis de la façon suivante :

$$\begin{cases} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad \textit{pour } n \geq 2 \end{cases}$$

1. *Écrire la fonction récursive* **unsigned int fibonacci_r(unsigned int n)** *qui calcule le* $n^{ième}$ *nombre de Fibonacci.*

2. *Vérifier avec votre programme que le nombre d'appels à la fonction pour calculer* F_n *est exactement égal à* $2 * F_{n+1} - 1$.

3. *Écrire la fonction itérative* **unsigned int fibonacci_i(unsigned int n)** *qui calcule le* $n^{ième}$ *nombre de Fibonacci.*

▸ **Exercice 4.** ***Écriture binaire***
Supposons que l'on souhaite afficher l'entier positif n sous la forme binaire. Pour l'ordre naturel, c'est-à-dire les bits de poids forts à gauche, comme on le fait en base 10, pour $n = 13 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3$ *on affiche 1101.*

1. *Écrire la fonction récursive* **void affiche_gauche(unsigned int n)** *qui affiche l'entier n sous la forme binaire avec les bits de poids fort à gauche.*

2. *Écrire la fonction récursive* **void affiche_droite(unsigned int n)** *qui affiche l'entier n sous la forme binaire avec les bits de poids fort à droite.*

▸ **Exercice 5.** ***Matrices***
Soit la structure suivante :

```
typedef struct
{
  /* nombre de lignes */
  int n;
  /* nombre de colonnes */
  int m;
  /* pointeur vers les coefficients */
  double **tab;
}matrix;
```

et la fonction

```
matrice *nouvelle_matrice(unsigned int n, unsigned int m)
{
```

```
  matrix * mat = (matrix *) malloc(sizeof(matrix));
  if(!mat)/* erreur memoire */
    exit(-1);
  mat->n = n;
  mat->m = m;
  mat->tab = (double **) malloc (mat->n * sizeof(double *));
  if(!mat->tab)/* erreur memoire */
    exit(-1);
  for (i = 0 ; i < m; i++)
  {
    mat->tab[i] = (double *) malloc (mat->m * sizeof(double));
    if( !mat-tab[i] )/* erreur memoire */
      exit(-1);
  }
  for (i = 0 ; i < n; i++)
    for (j=0; j < m; j++)
      mat->tab[i][j] = 0.;
  return mat;
}
```

1. *Écrire la fonction* **void libere_matrice(matrice *m)**.
2. *Écrire la fonction* **void afficher_matrice(matrice m)** *qui affiche* **m** *sur la sortie standard.*
3. *Écrire la fonction* **matrice *saisir_matrice()** *qui crée une nouvelle matrice en fonction de valeurs saisies par l'utilisateur.*
4. *Écrire la fonction* **matrice *multiplication_matrice(matrice m1, double scalaire)** *qui renvoie une nouvelle matrice A telle que* $A = m1 * scalaire$.
5. *Écrire la fonction* **matrice *somme_matrice(matrice m1, matrice m2)** *qui renvoie une nouvelle matrice A telle que* $A = m1+m2$ *ou NULL si l'opération n'est pas possible.*
6. *Écrire la fonction* **matrice *produit_matrice(matrice m1, matrice m2)** *qui renvoie une nouvelle matrice A telle que* $A = m1*m2$ *ou NULL si l'opération n'est pas possible.*
7. *Instancier une matrice de rotation en deux dimensions. L'utiliser pour écrire la fonction* **void dessine_cercle_pointille(int x_centre, int y_centre, MlvType *x_var)** *qui dessine en pointillé un cercle de centre* **(x_centre, y_centre)** *dans une fenêtre identifiée par* **x_var**[(1)].

[(1)] Il s'agit de faire tourner un segment de droite ou un arc de cercle autour du centre.