Langage C : TD6

Exercice 1 :

1. Définir un type `Etudiant` contenant : un identifiant (un entier long), un nombre de notes (un entier) et un tableau de notes (des réels). Le nombre de notes peut varier selon les étudiants (selon la formation par exemple).

```
typedef struct{
    long id;
    int nb_notes;
    double *notes;
} Etudiant;
```

2. Alloue la mémoire pour stocker les informations d'un Etudiant ayant n notes.

```
Etudiant allocEtu(int n)
{
    Etudiant e;
    e.nb_notes = n;
    e.notes = NULL;
    e.notes = (double *)calloc(n, sizeof(double));
}
```

OU

```
void allocEtu(Etudiant *e, int n)
{
    e->nb_notes = n;
    e->notes = NULL;
    e->notes = (double *)calloc(n, sizeof(double));
}
```

3. Alloue la mémoire pour stocker les informations de n structures de type Etudiant, en demandant le nombre de notes pour chaque étudiant.

```
Etudiant *allocNEtu(int n)
{
    int i;
    Etudiant *t = NULL;
    t = (Etudiant *) calloc(n, sizeof(Etudiant));
    if (t != NULL)
    {
        for (i = 0 ; i < n ; i++)
        {
            printf("Combien de notes ?");
            scanf("%d", &t[i].nb_notes);
            t[i] = allocEtu(&t[i], t[i].nb_notes);
        }
    }

    return t;
}
```

4. Libère toute la mémoire allouée pour stocker les informations de n structures de type Etudiant.

```c
void freeEtu(Etudiant *t)
{
    for (int i = 0 ; i < n ; i++)
    {
        free(t[i].notes);
    }
}
```

Exercice 2 : Dans cet exercice on considère une liste chaînée simple dont chaque cellule (ou nœud) contient un réel. On définit cette liste chaînée via le type `Cellule` suivant :

```c
typedef struct cell{
    double v;
    struct cell *suiv;
} Cellule;
```

1. `void affiche(Cellule *l)`, d'affichage des valeurs contenues dans une telle liste accessible par son pointeur de tête.

```c
void affiche(Cellule *l)
{
    Cellule *c = l;
    while (c != NULL)
    {
        printf("%lf", c->v);
        c = c -> suiv;
    }
}
```

2. `Cellule *en_queue(Cellule *l, double v)` d'ajout d'une valeur v donnée en dernière position de la liste pointée par l.

```c
Cellule *en_queue(Cellule *l, double v)
{
    Cellule *p = NULL, *c = l;
    p = (Cellule *) malloc(sizeof(Cellule));

    p -> v = v;
    p -> suiv = null;

    if (l == null) return p;

    while (c -> suiv != null)
    {
        c = c -> suiv;
    }

    c -> suiv = p;

    return l;
}
```