❑ **Exercice 1** : *Nombre harshad*

Les questions sont *indépendantes*, et on pourra toujours utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée. On supposera toujours déjà incluses les librairies usuelles du langage C.

On dit qu'un entier strictement positif est un *nombre harshad* (ou nombre de Niven) lorsqu'il est divisible par la somme de ses chiffres dans une base donnée. Dans cet exercice, on s'intéresse aux nombres harshad en base 10. Par exemples, 48 est un nombre harshad puisque 48 est divisible par 12, de même 63 (divisible par 9) ou encore 190 (divisible par 10) sont aussi des nombres harshad. Par contre 28, ou encore 104 ne sont pas des nombres harshad.

1. Ecrire une fonction de signature `bool est_harshad(int n)` qui renvoie `true` si et seulement si `n` est un nombre harshad.

```c
bool est_harshad(int n)
{
    int sum = 0;
    int num = n;
    while (num != 0)
    {
        sum += (num % 10);
        num = num / 10;
    }
    return (n % sum == 0);
}
```

2. Ecrire une fonction `main` qui prend un argument en ligne de commande une chaine de caractère, la convertit en entier (avec la fonction `atoi`) puis affiche `true` si cet entier est harshad et `false` sinon. Si aucun argument n'est donné, on affiche un message d'erreur. Par exemple, voici des comportements attendus en supposant que l'exécutable s'appelle `harshad.exe`

```
./harshad.exe 42
true
./harsahd.exe 53
false
./harshad.exe
Utilisation : ./harshad.exe <entier positif>
```

```c
int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Utilisation : %s <entier> \n", argv[0]);
        return EXIT_FAILURE;
    }
    int n = atoi(argv[1]);
    if (est_harshad(n))
    {
        printf("true\n");
    }
    else
    {
        printf("false\n");
    }
    return EXIT_SUCCESS;
}
```

3. Ecrire une fonction `int* get_harshad(int limit, int *nb)` qui prend en argument un entier `limit`, et renvoie un tableau contenant les nombres de harshad inférieurs ou égaux à `limit`. De plus `*nb` doit contenir après appel le nombre de nombres harshad inférieurs ou égaux à `limit`. Par exemple comme les nombres harshad inférieurs ou égaux à 20 sont `{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20}` l'appel `get_harshad(20, &n)` renvoie un tableau contenant ces entiers et après l'appel `n` contient la taille de ce tableau donc `13`.

```c
int *get_harshad(int limit, int *size)
{
    *size = 0;
    for (int i = 1; i <= limit; i++)
    {
        if (est_harshad(i))
        {
            *size = *size + 1;
        }
    }
    int *harshad = malloc(sizeof(int) * (*size));
    int cpt = 0;
    for (int i = 1; i < limit; i++)
    {
        if (est_harshad(i))
        {
            harshad[cpt] = i;
            cpt++;
        }
    }
    return harshad;
}
```

4. On recherche à présent des suites de nombres consécutifs qui sont tous harshad, par exemple les nombres 110, 111 et 112 sont 3 nombres harshad consécutifs. Dans un tableau quelconque de valeurs, on est donc amené à chercher le nombre maximal de valeurs consécutives. Ecrire une fonction de signature `int consecutifs(int tab[], int size, int *start)` qui renvoie le nombre maximal de valeurs consécutives dans le tableau `tab` de taille `size` et met à jour `*start` afin qu'elle contienne la première de ces valeurs. Par exemple cette fonction sur le tableau `{2, 7, 8, 13, 14, 15, 17, 18, 21}` renvoie 3 et `*start` vaut 13. En effet il y a un maximum de 3 valeurs consécutives et la première est 13.

```c
int consecutifs(int tab[], int size, int *start)
{
    int cpt = 1;
    int maxc = 1;
    int maxs = 1;
    for (int i = 11; i < size; i++)
    {
        if (tab[i] == tab[i - 1] + 1)
        {
            cpt += 1;
            if (cpt > maxc)
            {
                maxc = cpt;
                maxs = tab[i - cpt + 1];
            }
        }
        else
        {
            cpt = 1;
        }
    }
    *start = maxs;
    return maxc;
}
```

5. Rechercher le nombre maximal de nombres harshad consécutifs parmi ceux qui sont inférieurs à un million *et strictement supérieurs à 9*, combien il y en a-t-il et quel est le premier ce cette série ?

   En utilisant les fonctions précédents, on trouve qu'il y a au maximum 5 nombres harshad consécutifs dans l'intervalle considéré, le premier d'entre eux est 131 052.