TP N° 3– Chaînes de caractères

Vous **devez** utiliser l'outil `make` pour compiler vos programmes. Les options `-Wall -W -Werror` devront être utilisées.

Le but de ce TP est de s'entraîner à la manipulation de chaînes de caractères en réécrivant un certain nombre de fonctions disponibles dans `string.h`.

Pour mémoire, une chaîne de caractères correspond en fait à un tableau d'octets (caractères) dont la fin est marquée par l'octet 0 (qui représente la séquence d'échappement ASCII `'\0'`). La chaîne est généralement repérée par l'adresse de la première case de ce tableau. On utilise donc généralement le type `char *` pour la représenter.

Exercice 1 : Longueur d'une chaîne

En utilisant uniquement les « notations tableaux », écrire une fonction retournant la longueur d'une chaîne de caractères (nombre de caractères contenues dans la chaîne sans le marqueur de fin), dont le prototype est :

```
int mon_strlen(char s[]);
```

Exercice 2 : Longueur d'une chaîne (2)

En utilisant uniquement les « notations pointeurs » réécrire la même fonction, dont le prototype devient :

```
int mon_strlen(const char *s);
```

Exercice 3 : Comparaison de chaînes

Écrire une fonction comparant 2 chaînes de caractères `s1` et `s2` selon l'ordre lexicographique, dont le prototype est :

```
int mon_strcmp(const char * s1, const char * s2);
```

Cette fonction retourne un entier :
— inférieur à 0 si `s1 < s2` ;
— égal à 0 si `s1 = s2` ;
— supérieur à 0 si `s1 > s2`.

Exercice 4 : Comparaison de chaînes (2)

Écrire une fonction similaire à la fonction `mon_strcmp` mais n'opérant que sur les n premiers caractères (au plus) des 2 chaînes, dont le prototype est :

```
int mon_strncmp(const char * s1, const char * s2, int n);
```

Exercice 5 : Concaténation de chaînes

Écrire une fonction concaténant la chaîne `s2` au bout de la chaîne `s1`, dont le prototype est :

```
char *mon_strcat(char *s1, const char *s2);
```

Cette fonction retourne `s1` (terminée par `'\0'`), `s1` est supposée assez grande pour permettre de réaliser l'opération.

Exercice 6 : Chercher un caractère..

Écrire une fonction qui retourne un pointeur sur la première occurrence d'un caractère dans une chaîne ou `NULL` si le caractère n'a pas été trouvé. Le prototype de cette fonction est :

```
char *mon_strchr(char *s, int c);
```

Exercice 7 : Chercher une aiguille..

Écrire une fonction cherchant la première occurence d'une sous-chaîne (`needle`*) dans une chaîne de référence (`haystack`†). Cette fonction retourne l'adresse du premier caractère de la sous-chaîne ou `NULL` si la sous-chaîne n'est pas trouvée. Le prototype de la fonction est :

```
char *mon_strstr(char *haystack, char *needle);
```

Exercice 8 : Réutilisation

Réécrire la fonction `mon_strstr` en utilisant intelligemment les autres fonctions définies dans cet exercice (comme `mon_strncmp`, etc.)

∗. en français needle veut dire aiguille
†. en français haystack veut dire botte de foin