Exercices sur les chaînes de caractères

Dans ce TD/TP, on écrit des versions personnelles des fonctions de `<string.h>` ainsi que les fonction de test associées.

En TP, organiser le code en 5 fichiers `MyString.h`, `MyString.c`, `MyStringTest.h`, `MyStringTest.c`, et `Main.c`, à l'intérieur d'un répertoire `MyString`. Rajouter une fonction `MyStringTest_runAll()` qui lance tous les tests, et invoquer cette fonction dans `Main.c`. Écrire également un fichier `Makefile` pour compiler ce programme.

Exercice 1. On rappelle que `strlen()` calcule le nombre de bytes dans une chaîne.

```
$ man 3 strlen
```

```
SYNOPSIS
       #include <string.h>
       size_t strlen(const char *s);
DESCRIPTION
       The  strlen() function calculates the length of the string s, excluding
       the terminating null byte ('\0').
RETURN VALUE
       The strlen() function returns the number of bytes in the string s.
```

En écrire une version personnelle `MyString_strlen()`, avec une boucle `while`.

```
size_t MyString_strlen (char const string[]);
```

Le test suivant doit passer :

```
void MyStringTest_strlen (void) {
  assert (MyString_strlen ("banana") == 6);
  assert (MyString_strlen ("x") == 1);
  assert (MyString_strlen ("") == 0);
}
```

Exercice 2. On rappelle que `strchr()` et `strrchr()` recherchent un caractère dans une chaîne.

```
$ man 3 strchr
```

```
SYNOPSIS
       #include <string.h>
       char *strchr (const char *s, int c);
       char *strrchr(const char *s, int c);
DESCRIPTION
       The  strchr() function returns a pointer to the first occurrence of the
       character c in the string s.

       The strrchr() function returns a pointer to the last occurrence of  the
       character c in the string s.

       Here "character" means "byte"; these functions do not work with wide or
       multibyte characters.
RETURN VALUE
       The strchr() and strrchr() functions return a pointer  to  the  matched
       character  or NULL if the character is not found.  The terminating null
       byte is considered part of the string, so that if  c  is  specified  as
       '\0', these functions return a pointer to the terminator.
```

En écrire les versions personnelles suivantes, ainsi que les tests associés.

```
char * MyString_strchr  (char const string[], int character);
char * MyString_strrchr (char const string[], int character);
```

```
void MyStringTest_strrchr (void);
void MyStringTest_strchr  (void);
```

Exercice 3. On rappelle que `strpbrk()` recherche dans une chaîne la première occurrence d'un caractère appartenant à un ensemble de caractères donné.

```
$ man 3 strpbrk
```

```
SYNOPSIS
       #include <string.h>
       char *strpbrk(const char *s, const char *accept);
DESCRIPTION
       The  strpbrk() function locates the first occurrence in the string s of
       any of the bytes in the string accept.
RETURN VALUE
       The strpbrk() function returns a pointer to the byte in s that  matches
       one of the bytes in accept, or NULL if no such byte is found.
```

En écrire la version personnelle suivante, ainsi que le test associé.

```
char * MyString_strpbrk (char const string[], char const accept[]);
```

```
void MyStringTest_strpbrk (void);
```

Exercice 4. On rappelle que `strcmp()` et `strncmp()` comparent deux chaînes lexicographiquement. Les caractères sont simplement comparés via leurs représentations numériques.

```
$ man 3 strcmp
```

```
SYNOPSIS
       #include <string.h>
       int strcmp (const char *s1, const char *s2);
       int strncmp(const char *s1, const char *s2, size_t n);
DESCRIPTION
       The  strcmp()  function compares the two strings s1 and s2.  It returns
       an integer less than, equal to, or greater than zero if  s1  is  found,
       respectively, to be less than, to match, or be greater than s2.

       The  strncmp()  function  is similar, except it compares only the first
       (at most) n bytes of s1 and s2.

RETURN VALUE
       The strcmp() and strncmp() functions return an integer less than, equal
       to, or greater than zero if s1 (or the first n bytes thereof) is found,
       respectively, to be less than, to match, or be greater than s2.
```

En écrire des versions personnelles, ainsi que les tests associés.

```
int MyString_strcmp  (char const string1[], char const string2[]);
int MyString_strncmp (char const string1[], char const string2[], size_t n);
```

```
void MyStringTest_strcmp  (void);
void MyStringTest_strncmp (void);
```

On aura avantage à coder et réutiliser les fonctions auxiliaires suivantes comparant un caractère et calculant la longueur du préfixe commun de deux chaînes.

```
int    MyString_compareChar            (char char1, char char2);
size_t MyString_commonPrefixLength     (char const s1[], char const s2[]);
size_t MyString_commonPrefixLengthUpTo (char const s1[], char const s2[], size_t n);
```

Tester chacune des fonctions auxiliaires :

```
void MyStringTest_compareChar            (void);
void MyStringTest_commonPrefixLength     (void);
void MyStringTest_commonPrefixLengthUpTo (void);
```

Exercice 5. On rappelle que `strstr()` recherche une sous-chaîne dans une chaîne.

```
$ man 3 strstr
```

```
SYNOPSIS
       #include <string.h>
       char *strstr(const char *haystack, const char *needle);
DESCRIPTION
       The  strstr() function finds the first occurrence of the substring nee-
       dle in the string haystack.  The terminating null bytes ('\0') are  not
       compared.
RETURN VALUE
       This  function returns a pointer to the beginning of the  located  sub-
       string, or NULL if the substring is not found.
```

En écrire une version personnelle, ainsi que le test associé :

```
char * MyString_strstr (char const string[], char const substring[]);
```

```
void MyStringTest_strstr (void);
```

Exercice 6. On rappelle que `strcpy()` et `strncpy()` copient une chaîne dans une autre.

```
$ man 3 strcpy
```

```
SYNOPSIS
       #include <string.h>
       char *strcpy (char *dest, const char *src);
       char *strncpy(char *dest, const char *src, size_t n);
DESCRIPTION
       The  strcpy()  function  copies the string pointed to by src, including
       the terminating null byte ('\0'), to the buffer  pointed  to  by  dest.
       The  strings  may  not overlap, and the destination string dest must be
       large enough to receive the copy.  Beware  of  buffer  overruns!

       The  strncpy()  function is similar, except that at most n bytes of src
       are copied.  Warning: If there is no null byte among the first n  bytes
       of src, the string placed in dest will not be null-terminated.

       If  the  length of src is less than n, strncpy() writes additional null
       bytes to dest to ensure that a total of n bytes are written.
RETURN VALUE
       The strcpy() and strncpy() functions return a pointer to  the  destina-
       tion string dest.
```

En écrire des versions personnelles, ainsi que les tests associés :

```
char * MyString_strcpy  (char target[], char const source[]);
char * MyString_strncpy (char target[], char const source[], size_t n);
```

```
void MyStringTest_strcpy  (void);
void MyStringTest_strncpy (void);
```

Exercice 7. On rappelle que `strcat()` et `strncat()` concatènent une chaîne à la suite d'une autre.

```
$ man 3 strcat
```

```
SYNOPSIS
       #include <string.h>
       char *strcat (char *dest, const char *src);
       char *strncat(char *dest, const char *src, size_t n);
DESCRIPTION
       The  strcat() function appends the src string to the dest string, over-
       writing the terminating null byte ('\0') at the end of dest,  and  then
       adds  a  terminating  null  byte.  The strings may not overlap, and the
       dest string must have enough space for the  result.   If  dest  is  not
       large  enough, program behavior is unpredictable; buffer overruns are a
       favorite avenue for attacking secure programs.

       The strncat() function is similar, except  that  it  will  use at  most
       n  bytes  from  src, and  src  does  not  need  to  be  null-terminated
       if it contains n or more bytes. As with  strcat(), the resulting string
       in dest is always null-terminated.

       If  src  contains  n  or more bytes, strncat() writes n+1 bytes to dest
       (n  from  src plus the terminating null byte).  Therefore, the size  of
       dest must be at least  strlen(dest)+n+1.
RETURN VALUE
       The strcat() and strncat() functions return a pointer to the  resulting
       string dest.
```

En écrire des versions personnelles, ainsi que les tests associés :

```
char * MyString_strcat  (char target[], char const source[]);
char * MyString_strncat (char target[], char const source[], size_t n);
```

```
void MyStringTest_strcat  (void);
void MyStringTest_strncat (void);
```

Exercice 8. On rappelle que `strspn()` et `strcspn()` calculent le *span* et le *co-span* d'une chaîne, c-à-d la longueur du préfixe dont les caractères sont respectivement à l'intérieur et à l'extérieur d'un ensemble de caractère donné.

```
$ man 3 strspn
```

```
SYNOPSIS
       #include <string.h>
       size_t strspn (const char *s, const char *accept);
       size_t strcspn(const char *s, const char *reject);
DESCRIPTION
       The  strspn()  function calculates the length (in bytes) of the initial
       segment of s which consists entirely of bytes in accept.

       The strcspn() function calculates the length of the initial segment  of
       s which consists entirely of bytes not in reject.
RETURN VALUE
       The  strspn()  function returns the number of bytes in the initial seg-
       ment of s which consist only of bytes from accept.

       The strcspn() function returns the number of bytes in the initial  seg-
       ment of s which are not in the string reject.
```

En écrire des versions personnelles, ainsi que les tests associés :

```
size_t MyString_strspn  (char const string[], char const accept[]);
size_t MyString_strcspn (char const string[], char const reject[]);
```

```
void MyStringTest_strspn  (void);
void MyStringTest_strcspn (void),
```

Exercice 9. Les fonctions `strdup()` et `strndup` sont des fonctions POSIX (mais non ISO) permettant dupliquer une chaîne. La copie est allouée dynamiquement par la fonction, contrairement `strcpy()` qui écrit la copie dans un tableau déjà alloué par l'utilisateur.

```
$ man 3 strdup
```

```
SYNOPSIS
       #define _POSIX_C_SOURCE 200809L
       #include <string.h>
       char *strdup  (const char *s);
       char *strndup (const char *s, size_t n);
DESCRIPTION
       The strdup() function returns a pointer to a  new  string  which  is  a
       duplicate  of the string s.  Memory for the new string is obtained with
       malloc(3), and can be freed with free(3).

       The strndup() function is similar, but copies at most n bytes.  If s is
       longer  than  n,  only  n bytes are copied, and a terminating null byte
       ('\0') is added.
RETURN VALUE
       On success, the strdup() function returns a pointer to  the  duplicated
       string.   It  returns  NULL  if insufficient memory was available, with
       errno set to indicate the cause of the error.
```

En écrire des versions personnelles, ainsi que les tests associés :

```
char * MyString_strdup  (char const string[]);
char * MyString_strndup (char const string[], size_t max_length);
```

```
void MyStringTest_strdup  (void);
void MyStringTest_strndup (void);
```

Exercice 10. Écrire une fonction `MyString_trim()` qui renvoie la copie d'une chaîne après élimination des blancs en début et en fin de chaîne. La copie est allouée dynamiquement. L'ensemble des blancs à éliminer est passé en deuxième argument.

Écrire la fonction de test associée.

```
char * MyString_trim (char const string[], char const blanks[]);
```

```
void MyStringTest_trim  (void);
```

Exercice 11. La fonction `memcmp()` compare lexicographiquement au plus `n` premiers bytes de deux zones de mémoire.

```
SYNOPSIS
       #include <string.h>
       int memcmp(const void *s1, const void *s2, size_t n);
DESCRIPTION
       The  memcmp()  function compares the first n bytes (each interpreted as
       unsigned char) of the memory areas s1 and s2.
RETURN VALUE
       The memcmp() function returns  an  integer  less  than,  equal  to,  or
       greater than zero if the first n bytes of s1 is found, respectively, to
       be less than, to match, or be greater than the first n bytes of s2.

       For a nonzero return value, the sign is determined by the sign  of  the
       difference  between  the  first  pair of bytes (interpreted as unsigned
       char) that differ in s1 and s2.
```

En écrire une version personnelle, ainsi que le test associé :

```
int MyString_memcmp (const void * memory1, const void * memory2, size_t n);
```

```
void MyStringTest_memcmp (void);
```

Exercice 12. La fonction `memset()` initialise les `n` premiers bytes d'une zone mémoire déjà allouée avec la valeur d'un byte donné.

```
SYNOPSIS
       #include <string.h>
       void *memset(void *s, int c, size_t n);
DESCRIPTION
       The  memset()  function  fills  the  first  n  bytes of the memory area
       pointed to by s with the constant byte c.
RETURN VALUE
       The memset() function returns a pointer to the memory area s.
```

En écrire une version personnelle, ainsi que le test associé :

```
void * MyString_memset (void * memory, int value, size_t n);
```

```
void MyStringTest_memset (void);
```