Exercice 1 : Tableaux

Pour les exemples ci-dessous, indiquez successivement s'ils compilent sans erreur et le cas échéant, si les chaînes de caractères sont modifiables ou non.

(a)
```
char s[] = 'foo';
```

Erreur, guillemet simple utilisé pour une chaîne de caractères.

(b)
```
char s[42] = "Les carottes sont cuites.";
```

L'expression compile et la chaîne est modifiable.

(c)
```
char s[2] = "Babel Fish";
```

Erreur : la taille de la chaîne est trop courte.

(d)
```
char s[] = "foo\0bar\0";
```

L'expression compile et la chaîne est modifiable. Elle vaut « foo ».

(e)
```
char *s = "Prix Béton 1981";
```

L'expression compile mais la chaîne est non modifiable.

(f)
```
char *s[] = {"foo", "bar", "baz"};
```

L'expression compile mais les chaînes sont non modifiables.

(g)
```
char s[] = {"foo" "bar" "baz"};
```

L'expression compile et la chaîne est modifiable. Elle vaut « foobarbaz ».

Exercice 2 : Manipulation des chaînes de caractères

Que retournent chacunes de ces fonctions ?

(a)
```
int foo() { char s[] = "foo"; return strlen(s); }
```

3

(b)
```
int foo() { char s[] = "foo"; return sizeof(s); }
```

4

(c)

```
int foo() { char s[] = "foo"; return s[3]; }
```

> 0 (caractère nul)

(d)

```
int foo() { char s[] = "foo"; return s[4]; }
```

> Indéterminé, on accède à une zone mémoire qui n'est pas allouée pour la chaîne. Une erreur de segmentation peut survenir.

(e)

```
int foo() { char s[] = "foo\n\t\0bar"; return strlen(s); }
```

> 5 (caractère nul après le $\widehat{\ }$)

Exercice 3

En utilisant la bibliothèque *ctype.h*, écrire une fonction qui transforme une chaîne de caractères reçue en majuscules. Le prototype de la fonction est `void majuscule(char *s)`.

```
void majuscule(char *s) {
  while (*s) {
    *s = toupper(*s);
    s++;
  }
}
```

Exercice 4

Soit un texte d'entrée, indiquer la position à laquelle se trouve une sous-chaîne, si elle existe. Sinon retourne -1. Le prototype de la fonction est `int position(char s[], char sub[])`. Aidez-vous également de la bibliothèque *ctype.h*.

```
int position(char s[], char sub[]) {
  if (strstr(s, sub) == NULL) {
    return -1;
  }
  return strstr(s, sub) - s;
}
```

Exercice 5

Que fait cette fonction et que retourne-t-elle ?

```
int mysterious()
{
    char v[] = "aeiouy";
    char s[] = "Hello, world!";
    size_t c = 0, k = 0;
    while (s[k] != '\0')
    {
```

```
        int n = 0;
        while (v[n] != '\0')
            c += s[k] == v[n++];
        k++;
    }
    return c;
}
```

La fonction compte le nombre de voyelles dans la chaîne `s`. Elle retourne la valeur 3.

Exercice 6 : Problème Difficile

Il existe trois types de modifications qui peuvent être appliquées à une chaîne de caractère : insérer un caractère, supprimer un caractère, ou remplacer un caractère. Considérant deux chaînes de caractères, écrivez une fonction qui retourne si les deux chaînes ne sont qu'à une modification près.

— `pale, ple -> true`
— `pales, pale -> true`
— `pale, bale -> true`
— `pale, bake -> false`

```
bool one_away(char s1[], char s2[])
{
    int i = 0, j = 0, k = 0;
    while (s1[i] != '\0' && s2[j] != '\0')
    {
        if (s1[i] == s2[j])
        {
            i++;
            j++;
        }
        else if (s1[i] != s2[j])
        {
            if (s1[i + 1] == s2[j] || s1[i] == s2[j + 1])
            {
                i++;
                j++;
            }
            else
            {
                return 0;
            }
        }
    }
    if (s1[i] == '\0' && s2[j] == '\0')
    {
        return 1;
    }
    else if (s1[i] == '\0')
    {
        while (s2[j] != '\0')
        {
            if (s2[j + 1] == s2[j])
            {
                return 0;
            }
            j++;
        }
        return 1;
    }
    else if (s2[j] == '\0')
    {
        while (s1[i] != '\0')
        {
            if (s1[i + 1] == s1[i])
            {
                return 0;
            }
            i++;
        }
        return 1;
    }
    return 0;
}
```