Partie II

Les programmes des exercices qui suivent doivent être écrits en langage C. L'annexe vous rappelle les prototypes de quelques fonctions de la bibliothèque standard du langage C.

Exercice 1 :

Dans cet exercice, **vous n'avez pas le droit d'utiliser les fonctions de la bibliothèque standard**. Les fonctions demandées aux questions Q1, Q2, Q3 et Q4 doivent être sans saisie et sans affichage.

Q1. Ecrire une fonction qui calcule et renvoie la longueur d'une chaîne de caractères.

```
size_t my_strlen(const char *str){
    size_t l = 0;
    while (str[l] != '\0'){
        ++l;
    }
    return l;
}
```

Remarque : dans la table des codes ASCII, les codes ASCII des chiffres « se suivent » :
- le code ASCII du chiffre '0' est égal à 48 en décimal (0x30 en hexadécimal) ;
- le code ASCII du chiffre '1' est égal à 49 en décimal (0x31 en hexadécimal) ;
- le code ASCII du chiffre '2' est égal à 50 en décimal (0x32 en hexadécimal) ;
- ...
- le code ASCII du chiffre '9' est égal à 57 en décimal (0x39 en hexadécimal) ;

Q2. Ecrire une fonction qui convertit une chaîne de caractères numérique en entier. La chaîne de caractères reçue en paramètre représente un entier écrit en décimal. Les éventuels espaces placés en début de chaîne et les éventuels caractères non valides situés en fin de chaîne doivent être ignorés.

Exemples :

Chaîne de caractères à convertir	*Résultat*
" 456"	456
"abc"	0
"-0"	0
"-+15"	0
"-17hello "	-17
" +305test"	305

```
int my_atoi(const char *s){
    size_t i = 0;
    int n = 0;
    bool pos = true;
    while(s[i] == ' '){
        ++i;
    }
    if (s[i] == '-'){
        pos = false;
        ++i;
    }
    else if (s[i] == '+'){
        ++i;
    }
    while (s[i] >= '0' && s[i] <= '9'){
        n = 10*n + s[i] - '0';
        ++i;
    }
    if (!pos){
        n = -n;
    }
    return n;
}
```

Q3. Ecrire une fonction qui convertit un entier en chaîne de caractères qui le représente en décimal. La fonction reçoit notamment en paramètre l'adresse du premier élément d'un tableau de caractères alloué dans la fonction appelante. On supposera que ce tableau a toujours une taille suffisante pour pouvoir recevoir le résultat de la conversion.

```
void int_to_string(int n, char *t){
    if (n < 0){
        n = -n;
        *t = '-';
        ++t ;
    }
    size_t i = 0;
    do{
        t[i] = n % 10 + '0';
        ++i;
        n /= 10;
    }while(n);
    t[i] = '\0';
    for (size_t j = 0; j < i/2; ++j){
        char temp = t[j];
        t[j] = t[i - j - 1];
        t[i - j - 1] = temp;
    }
}
```

Q4. Ecrire une fonction qui recherche la première occurrence d'une sous-chaîne *needle* dans une chaîne *haystack*. Les caractères '\0' ne sont pas comparés. Cette fonction renvoie un pointeur sur le

début de la sous-chaîne *needle* (dans la chaîne *haystack*), ou NULL si la sous-chaîne *needle* n'est pas trouvée.

```
const char *my_strstr(const char *haystack, const char *needle){
    size_t l1 = my_strlen(haystack);
    size_t l2 = my_strlen(needle);
    if (l2 > l1){
        return NULL;
    }
    for (size_t i = 0; i <= l1 - l2; ++i){
        bool find = true;
        for(size_t j = 0; j < l2 && find; ++j){
            if (haystack[i+j] != needle[j]){
                find = false;
            }
        }
        if (find){
            return haystack+i;
        }
    }
    return NULL;
}
```

Q5. Ecrire une fonction main() la plus simple possible qui appelle la fonction définie à la question Q3.

```
int main(){
    char tab[30]; //surdimensionné
    int_to_string(12345, tab);
    return 0;
}
```

Exercice 2 : tableau d'entiers

Q1. Ecrire une fonction qui renvoie un entier aléatoire compris entre deux bornes B1 et B2 incluses (c'est-à-dire dans l'intervalle [B1 ; B2]). Les bornes B1 et B2 seront transmises en paramètre à la fonction.

N. B. : Vous utiliserez la fonction **`rand()`** qui renvoie un entier pseudo-aléatoire compris entre 0 et RAND_MAX, bornes incluses (c'est-à-dire dans l'intervalle [0 ; RAND_MAX]). RAND_MAX est une constante dont la valeur peut varier d'un compilateur à l'autre, mais elle est forcément d'au moins 32767. On supposera que (B2 – B1) est toujours inférieur ou égal à RAND_MAX.

```
int aleatoire(int B1, int B2){
    if (B1 > B2){
        int temp = B1;
        B1 = B2;
        B2 = temp;
    }
    return rand() %(B2 - B1 + 1) + B1;
}
```

Q2. Ecrire une fonction qui initialise un tableau d'entiers avec des nombres aléatoires compris entre deux bornes B1 et B2 incluses. Les bornes B1 et B2 seront notamment transmises en paramètre à la fonction.

```
void init (int *t, size_t size, int B1, int B2){
    for (size_t i = 0; i < size; ++i){
        t[i] = aleatoire(B1, B2);
    }
}
```

Q3. Ecrire une fonction qui compte le nombre de valeurs d'un tableau d'entiers égale à une valeur passée en paramètre. Cette fonction doit également permettre de récupérer l'indice de la première occurrence de la valeur recherchée. Si la valeur recherchée n'est pas trouvée, l'indice récupéré doit être égale à la taille du tableau. Cette fonction doit être sans saisie et sans affichage.

```
size_t counti(const int *t, size_t size, int value, size_t *pind){
    *pind = size;
    size_t cpt = 0;
    for(size_t i = 0; i < size; ++i){
        if (t[i] == value){
            ++cpt;
            if (*pind == size){
                *pind = i;
            }
        }
    }
    return cpt;
}
```

Q4. Ecrire une version récursive de la fonction de la question précédente. Cette fonction ne doit pas utiliser de boucle for, while ou do while.

```
// Avant le premier appel de cette fonction, il faut initialiser la valeur
// située à l'adresse pind avec la taille du tableau.
// C'est fait par la fonction nommée countr.
size_t countrec(const int *t, size_t size, int value, size_t *pind){
    if (size == 0){
        return 0;
    }
    if (t[size - 1] == value){
        *pind = size -1;
        return 1 + countrec(t, size-1, value, pind);
    }
    return countrec(t, size-1, value, pind);
}

size_t countr(const int *t, size_t size, int value, size_t *pind){
    *pind = size;
    return countrec(t, size, value, pind);
}
```

Q5. Ecrire la définition d'une fonction qui trie dans l'ordre croissant un tableau d'entiers suivant l'algorithme « tri par sélection ».

```
void tri_selection(int *tab, size_t size){
    size_t ind_min;
    int val_min;

    for (size_t i = 0 ; i+1 < size ; ++i){
        ind_min = i;
        val_min = tab[i];
        for (size_t j = i + 1 ; j < size ; ++j){
            if (tab[j] < val_min){
                ind_min = j ;
                val_min = tab[j];
            }
        }
        tab[ind_min] = tab[i];
        tab[i] = val_min;
    }
}
```

Q6. Ecrire une définition main qui appelle les fonctions définies aux questions Q2 et Q3. La valeur recherchée sera saisie par l'utilisateur, et les résultats seront affichés sur la sortie standard.

```
int main(){
    int tab[100];
    init(tab, 100, 10, 20);
    int val;
    printf("Saisir une valeur à rechercher : ");
    scanf("%d", &val);
    size_t ind;
    size_t cpt = counti(tab, 100, val, &ind);
    printf("La valeur %zu est présente %zu fois.\n", val, cpt);
    if (cpt > 0){
        printf("La première occurrence est située à l'indice %zu.\n", ind);
    }
    return 0;
}
```

Annexe : prototypes de fonctions de la bibliothèque standard

```
int rand(void);
void srand(unsigned int seed);
int printf(const char *format, ...);
int scanf(const char *format, ...);
```