Exercice 1 : Choix multiples

(a) Cocher les propositions vraies concernant `strncmp` ?

A. La fonction compare deux chaînes de caractères jusqu'à un certain nombre de caractères.
B. La fonction compare deux chaînes de caractères jusqu'à la fin.
C. La fonction retourne 1 si les chaînes sont égales.
D. La fonction peut s'appeler avec `strncmp("abc", "ab", 2)`.

(a) **A, D**

(b) Quel est le rôle de l'opérateur^ ?

A. Puissance comme dans `103`.
B. Ou exclusif.
C. Ou inclusif.
D. Et logique.
E. Modulo.
F. Inversion bit à bit.

(b) **B**

(c) Quelles sont les déclarations de tableaux valides ?

A. `int a[] = 0;`
B. `int a[10];`
C. `int a[10] = {1, 2, 3, 4, 5};`
D. `int a[] = {1, 2, 3, 4, 5};`
E. `int a[10] = 0;`
F. `int a[3]; a = {1,2,3};`

(c) **B, C, D**

(d) Cocher ce qui est vrai concernant `int *p` et `int p[10]` ?

A. `int *p` est un pointeur vers un entier, `int p[10]` est un tableau de 10 entiers.
B. La taille de `int *p` est la taille d'une adresse, la taille de `int p[10]` est la taille de 10 entiers soit 40 octets.
C. `int *p` est un tableau de 10 entiers, `int p[10]` est un pointeur vers un entier.
D. Les deux peuvent être passés à une fonction pour être modifiés.
E. L'accès à l'élément 2 s'écrit dans les deux cas `p[2]`.

(d) **A, B, D, E**

Exercice 2 : Fonctions

(a) Écrire une fonction `vowel` qui prend en paramètre une chaîne de caractère et qui retourne le nombre de voyelles dans la chaîne.

```
int vowel(char *str) {
    int count = 0;
    while (*str != '\0') {
        char c = tolower(*str);
        if (c == 'a' || c == 'e' || c == 'i' ||
            c == 'o' || c == 'u' || c == 'y')
            count++;
        str++;
    }
    return count;
}
```

(b) Écrire une fonction `max_interval` qui calcule l'intervalle le plus grand dans un tableau d'entiers passé en paramètre. Le second paramètre est la taille du tableau.

```
int max_interval(int *a, size_t size) {
    int max = 0;
    for (size_t i = 0; i < size; i++) {
        for (size_t j = i + 1; j < size; j++) {
            int interval = a[j] - a[i];
            if (interval > max)
                max = interval;
        }
    }
    return max;
}
```

(c) Écrire une fonction `is_palindrome` qui retourne vrai si une chaîne de caractères est un palindrome, faux sinon. La chaîne est passée en paramètre.

```
bool is_palindrome(char *str) {
    size_t size = strlen(str);
    for (size_t i = 0; i < size / 2; i++) {
        if (str[i] != str[size - i - 1])
            return false;
    }
    return true;
}
```

(d) Écrire une fonction qui prend en paramètre deux tableaux d'entiers et qui s'assure que l'un est bien dans l'ordre inverse que l'autre. La fonction prend en paramètre les deux tableaux et leur taille qui est commune.

```
bool is_reverse(int *a, int *b, size_t size) {
    for (size_t i = 0; i < size; i++) {
        if (a[i] != b[size - i - 1])
            return false;
    }
    return true;
}
```

(e) Écrire un une fonction qui rempli un tableau à deux dimensions de taille 10x10 avec des valeurs aléatoires entre 0 et 100. La fonction prend en paramètre le tableau, le nombre de lignes et le nombre de colonnes.

```
void fill_random(int a[10][10]) {
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++)
            a[i][j] = rand() % 101;
    }
}
```

Exercice 3 : Programmation

Écrire un programme complet qui lit sur les arguments `--min=X` et `--max=Y` où X et Y sont des entiers

positifs. Le programme appelle une fonction `compute(x, y)` avec les valeurs capturées sur les arguments. Cette fonction retourne vraie si elle s'est exécutée correctement, sinon faux.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

extern bool compute(int x, int y);

int main(int argc, char* argv[]) {
    int x, y;
    for (int arg = 1; arg < argc; arg++) {
        if (strncmp(argv[arg], "--min=", 6) == 0) {
            x = atoi(argv[arg + 1]);
            continue;
        }
        if (strncmp(argv[arg], "--max=", 6) == 0) {
            y = atoi(argv[arg + 1]);
            continue;
        }
        printf("Usage: %s --min=X --max=Y\n", argv[0]);
        return 1;
    }
    return compute(x, y);
}
```