Université Mohammed V
Ecole Nationale Supérieure de l'Informatique et de l'Analyse des Systèmes
Département Génie Logiciel

Correction du TP2 – Programmation en langage C

Travaux pratiques

Langage C (M1.1.2)
Semestre 1 – 2026–2027
Correction

2 Correction du TP2

Tableaux, chaînes et algorithmes de base

Pr. Youness BOUTYOUR youness.boutyour@um5.ac.ma

Version du 27 septembre 2026

Table des matières

Méthode de correction **2**

Exercice 1 – Statistiques, occurrences, inversion et tri 2

Exercice 2 – Tableau trié et tableau premier 2

Exercice 3 – Couples de nombres amicaux 3

Exercice 4 – Chaînes de caractères 4

Exercice 5 – Suppression d'occurrences 4

Exercice 6 – Fusion de tableaux triés 5

Exercice 7 – Valeurs distinctes 5

Méthode de correction

Les solutions séparent les parcours, les transformations et les tests de propriétés. Les indices commencent à zéro dans le code ; les doublons sont conservés lorsque l'énoncé demande une fusion et supprimés uniquement dans l'exercice 7. Compiler avec`gcc -std=c11 -Wall -Wextra -Wpedantic fichier.c`.

Exercice 1 – Statistiques, occurrences, inversion et tri

Un seul tableau est parcouru pour les statistiques ; les occurrences sont comptées pour chaque valeur distincte, puis le tableau est inversé et trié par sélection.

Correction complète

```c
#include <stdio.h>

static void afficher(int *t, int n) { for (int i = 0; i < n; ++i) printf("%d ", t[i]); putchar('\n'); }
static void trier(int *t, int n) {
    for (int i = 0; i < n - 1; ++i) for (int j = i + 1; j < n; ++j)
        if (t[j] < t[i]) { int v = t[i]; t[i] = t[j]; t[j] = v; }
}

int main(void) {
    int t[100], n, somme = 0;
    if (scanf("%d", &n) != 1 || n < 1 || n > 100) return 1;
    for (int i = 0; i < n; ++i) { if (scanf("%d", &t[i]) != 1) return 1; somme += t[i]; }
    int min = t[0], max = t[0];
    for (int i = 1; i < n; ++i) { if (t[i] < min) min = t[i]; if (t[i] > max) max = t[i]; }
    printf("min=%d max=%d moyenne=%.2f\n", min, max, (double) somme / n);
    for (int i = 0; i < n; ++i) { int deja = 0; for (int j = 0; j < i; ++j) if (t[j] == t[i]) deja = 1; if (!deja) { int c = 0; for (int j = 0; j < n; ++j) if (t[j] == t[i]) ++c; printf("%d:%d ", t[i], c); } }
    puts("\ninverse:"); for (int i = n - 1; i >= 0; --i) printf("%d ", t[i]);
    puts("\ntrie:"); trier(t, n); afficher(t, n);
    return 0;
}
```

Exercice 2 – Tableau trié et tableau premier

Le caractère trié est vérifié localement par comparaison de voisins ; la primalité de chaque élément est testée jusqu'à sa racine carrée.

Correction complète

```c
#include <stdio.h>

static int premier(int n) { if (n < 2) return 0; for (int d = 2; d <= n / d; ++d) if (n % d == 0) return 0; return 1; }
int main(void) {
    int t[100], n, trie = 1, tous_premiers = 1;
    if (scanf("%d", &n) != 1 || n < 1 || n > 100) return 1;
    for (int i = 0; i < n; ++i) { if (scanf("%d", &t[i]) != 1) return 1; if (i && t[i] < t[i - 1]) trie = 0; if (!premier(t[i])) tous_premiers = 0; }
    printf("trie=%s, tableau premier=%s\n", trie ? "oui" : "non", tous_premiers ? "oui" : "non");
    return 0;
}
```

Exercice 3 – Couples de nombres amicaux

Chaque paire d'indices distincts est examinée une seule fois et les sommes de diviseurs propres sont calculées par une fonction partagée.

Correction complète

```c
#include <stdio.h>

static int somme_diviseurs(int n) { int s = n > 1 ? 1 : 0; for (int d = 2; d <= n / d; ++d) if (n % d == 0) { s += d; if (d != n / d) s += n / d; } return s; }
int main(void) {
    int t[100], n;
    if (scanf("%d", &n) != 1 || n < 1 || n > 100) return 1;
    for (int i = 0; i < n; ++i) if (scanf("%d", &t[i]) != 1 || t[i] <= 0) return 1;
    for (int i = 0; i < n; ++i) for (int j = i + 1; j < n; ++j) if (t[i] != t[j] && somme_diviseurs(t[i]) == t[j] && somme_diviseurs(t[j]) == t[i]) printf("(%d,%d) ", t[i], t[j]);
    putchar('\n');
```

```
    return 0;
}
```

Exercice 4 – Chaînes de caractères

La comparaison, la recherche de sous-chaîne, la copie et la concaténation sont réalisées avec les fonctions standard et des tailles de destinations explicites.

Correction complète

```c
#include <stdio.h>
#include <string.h>

int main(void) {
    char s1[256], s2[256], s3[256], concat[512];
    if (!fgets(s1, sizeof s1, stdin) || !fgets(s2, sizeof s2, stdin)) return 1;
    s1[strcspn(s1, "\n")] = '\0'; s2[strcspn(s2, "\n")] = '\0';
    printf("comparaison = %d\n", strcmp(s1, s2));
    printf("sous-chaine = %s\n", strstr(s1, s2) ? "oui" : "non");
    strcpy(s3, s1); strcpy(concat, s1); strncat(concat, s2, sizeof concat - strlen(concat) - 1);
    printf("copie = %s\nconcat = %s\n", s3, concat);
    return 0;
}
```

Exercice 5 – Suppression d'occurrences

Les positions sont affichées avant la suppression ; la reconstruction compacte le tableau sans laisser de trou. Pour un tableau déjà trié, seule la zone égale à x doit être retirée.

Correction complète

```c
#include <stdio.h>

static void afficher(int *t, int n) { for (int i = 0; i < n; ++i) printf("%d ", t[i]); putchar('\n'); }
static int supprimer(int *t, int n, int x) { int e = 0; for (int i = 0; i < n; ++i) if (t[i] != x) t[e++] = t[i]; return e; }
```

```c
static int supprimer_trie(int *t, int n, int x) { int debut = 0; while (debut < n && t[debut] < x) ++debut; int fin = debut; while (fin < n && t[fin] == x) ++fin; for (int i = fin; i < n; ++i) t[debut++] = t[i]; return debut; }
int main(void) {
    int t[100], n, x, tableau_trie;
    if (scanf("%d", &n) != 1 || n < 1 || n > 100) return 1;
    for (int i = 0; i < n; ++i) if (scanf("%d", &t[i]) != 1) return 1;
    if (scanf("%d", &x) != 1) return 1;
    if (scanf("%d", &tableau_trie) != 1) tableau_trie = 0;
    printf("positions : "); for (int i = 0; i < n; ++i) if (t[i] == x) printf("%d ", i); putchar('\n');
    n = tableau_trie ? supprimer_trie(t, n, x) : supprimer(t, n, x); afficher(t, n);
    return 0;
    /* Pour un tableau deja trie, remplacer l'appel par : n = supprimer_trie(t,n,x); */
}
```

Exercice 6 – Fusion de tableaux triés

Deux indices avancent dans les tableaux d'entrée ; le plus petit élément disponible est recopié dans le tableau résultat, sans retri global.

Correction complète

```c
#include <stdio.h>

int main(void) {
    int a[100], b[100], c[200], n, m, i = 0, j = 0, k = 0;
    if (scanf("%d", &n) != 1 || n < 0 || n > 100) return 1;
    for (i = 0; i < n; ++i) if (scanf("%d", &a[i]) != 1) return 1;
    if (scanf("%d", &m) != 1 || m < 0 || m > 100) return 1;
    for (j = 0; j < m; ++j) if (scanf("%d", &b[j]) != 1) return 1;
    i = j = 0;
    while (i < n && j < m) c[k++] = a[i] <= b[j] ? a[i++] : b[j++];
    while (i < n) c[k++] = a[i++];
    while (j < m) c[k++] = b[j++];
    for (i = 0; i < k; ++i) printf("%d ", c[i]);
    putchar('\n');
    return 0;
}
```

Exercice 7 – Valeurs distinctes

Le tableau est compacté en conservant la première occurrence de chaque valeur et son ordre d'apparition.

Correction complète

```c
#include <stdio.h>

int main(void) {
    int t[100], n, taille = 0;
    if (scanf("%d", &n) != 1 || n < 0 || n > 100) return 1;
    for (int i = 0; i < n; ++i) {
        int deja = 0;
        if (scanf("%d", &t[i]) != 1) return 1;
        for (int j = 0; j < taille; ++j) if (t[j] == t[i]) deja = 1;
        if (!deja) t[taille++] = t[i];
    }
    for (int i = 0; i < taille; ++i) printf("%d ", t[i]);
    putchar('\n');
    return 0;
}
```