CORRECTION EXERCICES TABLEAUX
LANGAGE C (1 ESGI)

Programme 1

```
/*********************************************************************/
/*  Fonction : gestion de tableaux                                   */
/*  Date     : 15/12/2017                                            */
/*  Auteur   : F. Sananes                                            */
/*                                                                   */
/*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#define SIZE 50
// Définit une constante qui ne pourra pas varier tout au long du programme

int main(int argc, char **argv) {
    int answer;
    int array[SIZE];
    int dimension = 0;
    int positiveArray[SIZE];
    int negativeArray[SIZE];
    int temp;
    int i;
    int k;
    int j;

    do {
        printf("\n1 .. Saisie du tableau");
        printf("\n2 .. Affichage du tableau");
        printf("\n3 .. Inversion du tableau");
        printf("\n4 .. Répartition positifs et négatifs");
        printf("\n6 .. Finir");
        printf("\nChoix : ");
        scanf("%d", &answer);
        printf("\n");
        switch(answer) {
            case 1 :
                do {
                    printf("\nSaisir le nombre d'elements du tableau : ");
                    scanf("%u", &dimension);
                } while(dimension < 1 || dimension > SIZE);

                for(i = 0; i < dimension; i++) {
                    printf("element %d : ", i);
                    scanf("%d", &array[i]);
                }
                break;

            case 2 :
                if (dimension > 0 && dimension < SIZE)
                    for(i = 0; i < dimension; i++)
                        printf("\nelement %d : %d", i, array[i]);
                break;

            case 3 :
                if (dimension > 0 && dimension < 50) {
                    for(i = 0; i < dimension / 2; i++) {
                        temp = array[i];
                        array[i] = array[dimension - 1 - i];
                        array[dimension - 1 - i] = temp;
                    }
                }
                break;
```

```
            case 4 :
                j = 0;
                k = 0;

                /* répartition des nombres du tableau */
                for(i = 0; i < dimension; i++)
                    if (array[i] > 0) positiveArray[j++] = array[i];
                    else if (array[i] < 0) negativeArray[k++] = array[i];

                /* Affichage du résultat */
                if (j == 0) printf("\nTableau positif vide");
                else {
                    printf("\n\nTableau positif");
                    for(i = 0; i < j; i++)
                        printf("\nelement %d : %d", i, positiveArray[i]);
                }

                if (k == 0) printf("\n\nTableau négatif vide");
                else {
                    printf("\n\nTableau négatif");
                    for(i = 0; i < k; i++)
                        printf("\nelement %d : %d", i, negativeArray[i]);
                }
                break;
        }
    } while(answer != 6);
     return 0;
}
```

Programme 2

```
/*********************************************************************/
/*  Fonction : gestion de tableaux à deux dimensions                 */
/*  Date     : 15/12/2017                                            */
/*  Auteur   : F. Sananes                                            */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#define SIZE 50
// définit une constante qui ne pourra pas varier tout au long du programme

int main(int argc, char **argv) {
    int answer;
    int array[SIZE][SIZE];
    int sum = 0;
    unsigned linesNumber = 0;
    unsigned colsNumber = 0;
    int i;
    int j;

    do {
        printf("\n1 .. Saisie du tableau");
        printf("\n2 .. Affichage du tableau");
        printf("\n3 .. Calcul des sommes");
        printf("\n6 .. Finir");
        printf("\nChoix : ");
        scanf("%d", &answer);
        printf("\n");
        switch(answer) {
            case 1 :
                /* Saisie de la taille 1 : nb lignes*/
                do {
                    printf("\nSaisir le nombre de lignes du tableau : ");
                    scanf("%u", &linesNumber);
                } while(linesNumber <= 1 || linesNumber > SIZE);
```

```
                /* Saisie de la taille 2 : nb colonnes*/
                do {
                    printf("\nSaisir le nombre de colonnes du tableau : ");
                    scanf("%u", &colsNumber);
                } while(colsNumber <= 1 || colsNumber > SIZE);

                /* Saisie des éléments */
                for(i = 0; i < linesNumber; i++)
                    for(j = 0; j < colsNumber; j++) {
                        printf("\nelement (%d,%d) : ", i, j);
                        scanf("%d", &array[i][j]);
                    }
                break;

            case 2 :
                if (linesNumber > 0 && linesNumber < 50 &&
                        colsNumber > 0 && colsNumber < 50) {
                    /* Saisie des éléments */
                    for(i = 0; i < linesNumber; i++)
                        for(j = 0; j < colsNumber; j++) {
                            printf("\nelement (%d,%d) : %d", i, j, array[i][j]);
                        }
                }
                break;

                    case 3 :
                /* calcul des sommes de lignes*/
                for(i = 0; i < linesNumber; i++) {
                    printf("\nSomme ligne %d : ", i);
                    sum = 0;
                    for(j = 0; j < colsNumber; j++) sum += array[i][j];
                    printf("%d", sum);
                }

                /* calcul des sommes de colonnes*/
                for(i = 0; i < colsNumber; i++) {
                    printf("\nSomme colonne %d : ", i);
                    sum = 0;
                    for(j = 0; j < linesNumber; j++) sum += array[j][i];
                    printf("%d", sum);
                }
                break;
        }
    } while(answer != 6);

    return 0;
}
```

Programme 3

```
/*********************************************************************/
/*  Fonction : Comparaison de deux nombres                           */
/*  Date     : 15/12/2017                                            */
/*  Auteur   : F. Sananes                                            */
/*********************************************************************/

#include <stdio.h>

int main(int argc, char **argv) {
    int firstNumber;
    int secondNumber;
    int absFirstNumber;
    int absSecondNumber;
    int digitsFirstNumber[10];
    int digitsSecondNumber[10];
    int i = 0;
    int j = 0;
    int k = 0;
    int found = 1;
    int counters[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

```
    printf("Nombre 1 : ");
    scanf("%d", &firstNumber);
    printf("\nNombre 2 : ");
    scanf("%d", &secondNumber);

    /* Cet algo. a le mérite de compter les chiffres en même temps
          que la décomposition  */
    if(firstNumber < 0)
        absFirstNumber = -firstNumber;
    else absFirstNumber = firstNumber;
    if(secondNumber < 0)
        absSecondNumber = -secondNumber;
    else absSecondNumber = secondNumber;

    while(absFirstNumber > 0) {
        digitsFirstNumber[i] = absFirstNumber % 10;
        counters[digitsFirstNumber[i++]]++;
        absFirstNumber /= 10;
    }

    while(absSecondNumber > 0) {
        digitsSecondNumber[j] = absSecondNumber % 10;
        counters[digitsSecondNumber[j++]]--;
        absSecondNumber /= 10;
    }

    if (i == j) {
        found = 0;
        while(found == 0 && k < 10)
            if(counters[k++] != 0) found = 1;
    }

    if (found == 0)
        printf("\n%d et %d sont formés des mêmes chiffres",
               firstNumber, secondNumber);
      else printf("\n%d et %d ne sont pas formés des mêmes chiffres",
               firstNumber, secondNumber);
    return 0;
}
```

Programme 4

```
/*********************************************************************/
/*  Fonction : Linéarisation d'un tableau à deux dimensions          */
/*  Date     : 15/12/2017                                            */
/*  Auteur   : F. Sananes                                            */
/*********************************************************************/

#include <stdio.h>
#define SIZE 10

int main(int argc, char **argv) {
    int array2D[SIZE][SIZE];
    int array1D[SIZE * SIZE];
    int i;
    int j;
    int k = 0;
    unsigned linesNumber;
    unsigned colsNumber;
    int newSize;

    /* Saisie de la taille 1 : nb lignes*/
    do {
        printf("\nSaisir le nombre de lignes du tableau : ");
        scanf("%u", &linesNumber);
    } while(linesNumber <= 1 || linesNumber > SIZE);
```

```
    /* Saisie de la taille 2 : nb colonnes*/
    do {
        printf("\nSaisir le nombre de colonnes du tableau : ");
        scanf("%u", &colsNumber);
    } while(colsNumber <= 1 || colsNumber > SIZE);

    /* Saisie des éléments */
    for(i = 0; i < linesNumber; i++)
        for(j = 0; j < colsNumber; j++) {
            printf("\nelement (%d,%d) : ", i, j);
            scanf("%d", &array2D[i][j]);
        }

    /* linéarisation de t dans array1D */
    for(i = 0; i < linesNumber; i++)
        for(j = 0; j < colsNumber; j++)
            array1D[k++] = array2D[i][j];

    /* affichage de array1D */
    newSize = k;
    for(i = 0; i < newSize; i++)
        printf("\nelement : %d", array1D[i]);

    /* Recherche et suppression des 0 */
    for(i = 0; i < newSize; i++)
        if (array1D[i] == 0) {
            for(j = i; j < newSize - 1; j++)
                array1D[j] = array1D[j + 1];
            newSize--;
            i--;  /* i-- pour éviter les 0 successifs */
        }

    /* Affichage du résultat */
    printf("\n\nTableau resultat");
    for(i = 0; i < newSize; i++)
        printf("\nelement %d : %d", i, array1D[i]);
    return 0;
}
```

Programme 5

```
/*********************************************************************/
/*  Fonction : Exploitation d'un tableau de notes                    */
/*  Remarque : fonctionne si nb notes <24, sinon effectuer un        */
/*  modulo par 24                                                    */
/*  Date     : 15/12/2017                                            */
/*  Auteur   : F. Sananes                                            */
/*********************************************************************/
#include <stdio.h>
#include <conio2.h>
#define NB 12   /* sert à définir une constante */

int main(int argc, char **argv) {
    float points[NB];
    float max = 0;
    float min = 60;
    float avg = 0;
    int statsArray[7] = {0, 0, 0, 0, 0, 0, 0};
    int i = 0;
    int j = 0;
    int maxNumber = 0;
    clrscr();
    do {
        do {
            printf("Note %3d : ", i + 1);
            scanf("%f", &points[i]);
        } while(points[i] < 0. || points[i] > 20.);
        avg += points[i];
```

```
        if (points[i] > max) max = points[i];
        if (points[i] < min) min = points[i];

        if (points[i] <= 3) statsArray[0]++;
        else if (points[i] <= 6) statsArray[1]++;
        else if (points[i] <= 9) statsArray[2]++;
        else if (points[i] <= 14) statsArray[3]++;
        else if (points[i] <= 17) statsArray[4]++;
        else if (points[i] <= 19) statsArray[5]++;
        else statsArray[6]++;
    } while(++i < NB);
    avg /= NB;
    for(i = 0; i < 6; i++)
        printf("%d ", statsArray[i]);
    getch();
    printf(" %d\n", maxNumber);
    for(i = 0; i < 6; i++)
        if (statsArray[i] > maxNumber) maxNumber = statsArray[i];
    getch();
    printf("toto\n%d\n", maxNumber);
    getch();
    clrscr();
    for(i = maxNumber; i >= 1; i--) {
        gotoxy(1, maxNumber + 1 - i);
        printf("%2d >", i);
        getch();
    }
    gotoxy(5, maxNumber + 1);
    printf("+-------+-------+-------+-------+-------+-------+-------+");
    gotoxy(5, maxNumber + 4);
    printf("Max : %.2f  Min : %.2f  Moyenne : %.2f", max, min, avg);
    for(i = 0; i < 7; i++)
        for(j = 0; j < statsArray[i]; j++) {
            gotoxy(6 + 8 * i, maxNumber - j);
            printf("#######");
        }
    return 0;
}
```