


|   |   |   |
|---|---|---|
|  | <b>ING2 – Computer Systems Engineering</b><br><b>Test et Vérification - TD1</b> |   |
|   | <i>Subject : Informatique</i>   | <i>Date : Octobre 2023</i>                      |
|   |   | <i>Duration :</i><br><i>Number of pages : 4</i> |

## 1 Revue de code

### Exercice 1.

Pour chaque code suivant (Figures 2, 3 et 4) , vous devrez :

- Construire le Graphe de Flot de Contrôle.
- Identifier l'erreur.
- Si possible, trouver un CT qui exécute l'erreur mais qui provoque un problème qui ne se voit pas.
- Si possible, trouver un CT qui exécute l'erreur mais ne provoque pas problème.
- Si possible, trouver un CT qui n'exécute pas l'erreur.
- Corriger l'erreur et vérifier que le CT fournit maintenant le résultat attendu.

## 2 Un peu plus complexe

### Exercice 2.

Soit le code suivant :

```

1 public static int[] triBicolore(int[] t) {
2     int i; // start iterator
3     int izero; // last index for next zero (last box)
4     i = 1;
5     izero = t.length - 1;
6     while (i < izero) {
7         // if read a zero, swap with last index
8         if (t[i] == 0) {
9             t[i] = t[izero];
10            t[izero] = 0;
11            izero--;
12        }
13        i++;
14    }
15    return t;
16 }
```

Ce code réalise le tri suivant : pour un tableau composé aléatoirement de 0 et de 1, il place les 1 de manière contigüe, au début et les zéros à la fin.

- Dessinez le graphe de flot de contrôle de la fonction `triBicolore`.
- Déterminez **les cas de test** minimaux afin de couvrir toutes les instructions.

c. Ajouter les cas de test qui :

- n'exécute pas la boucle `while`,
- exécute **exactement une fois** la boucle `while`.

d. Indiquez la faute grossière mise en avant par la question précédente, la corriger.

e. Malgré la modification réalisée en question précédente, il subsiste une erreur, l'identifier.

f. Quel est le cas de test exécute l'erreur et provoque un pb qui se voit.

g. Proposez un cas de test qui exécute la faute mais qui ne se voit pas.

h. Proposez un cas de test qui n'exécute pas la faute.

### 3 Microsoft Zune

#### Exercise 3.

Zune est une gamme de balladeurs numériques destinés à concurrencer les iPods. Ces balladeurs ont été commercialisés de 2006 à 2011 en Amérique du Nord.

Le 31 décembre 2008, les lecteurs de première génération ont été touchés par un bug critique, les empêchant de démarrer.

Ce bug, corrigé par Microsoft, provient d'une partie du code des pilotes de gestion de la date (Figure 1), nous nous proposons de l'étudier.

```
1 | year = ORIGINYEAR;
2 | while (days > 365) {
3 |     if (IsLeapYear(year)) {
4 |         if (days > 366) {
5 |             days -= 366;
6 |             year += 1;
7 |         }
8 |     } else {
9 |         days -= 365;
10 |        year += 1;
11 |    }
12 | }
```

Figure 1: Code Zune

Le but de ce code est de calculer l'année en cours. En entrée, la variable "days" est égale au nombre de jours depuis le 1er janvier 1980, considéré comme le début du temps (chez Microsoft).

En sortie, la variable "year" est censée contenir le nombre correct d'années. (La constante ORIGINYEAR est fixée à 1980. Le prédicat IsLeapYear vérifie qu'une année est bissextile).

L'objectif de ce code est assez clair : il décrémente successivement le nombre de jours, de 365 pour les années courantes et de 366 pour les années bissextiles. À chaque fois, il incrémente le nombre d'années de 1.

À la fin du processus, le nombre d'années devrait avoir atteint la valeur actuelle correcte. Mais bien sûr, vous vous en doutez, tout ne s'est pas passé comme prévu, puisqu'il y a eu un bug. Pouvez-vous identifier le problème ?

a. Construire le Graphe de Flot de Contrôle.

b. Identifier l'erreur.

- c. Si possible, trouver un CT qui exécute l'erreur mais qui provoque un problème qui ne se voit pas.
- d. Si possible, trouver un CT qui exécute l'erreur mais ne provoque pas problème.
- e. Si possible, trouver un CT qui n'exécute pas l'erreur.
- f. Un collègue relit ce code et propose de rajouter un égal à la comparaison ligne 4. Repasser les CT et conclure sur la modification.
- g. Proposer une correction de l'erreur et vérifier que le CT fournit maintenant le résultat attendu.

```

1 public static int oddOrPos(int[] x) {
2     // Effects: if x==null throw NullPointerException
3     // else return the number of elements in x that
4     // are either odd or positive (or both)
5     int count = 0;
6
7     for (int i = 0; i < x.length; i++) {
8         if (x[i]%2 == 1 || x[i] > 0) {
9             count++;
10        }
11    }
12    return count;
13 }
14 // test: x=[-3, -2, 0, 1, 4]
15 // Expected = 3

```

Figure 2: Code 1

```

1 public static int findLast (int[] x, int y) {
2     // Effects: If x==null throw NullPointerException
3     // else return the index of the last element
4     // in x that equals y.
5     // If no such element exists, return -1
6     for (int i=x.length-1; i > 0; i--) {
7         if (x[i] == y) {
8             return i;
9         }
10    }
11    return -1;
12 }
13 // test: x=[2, 3, 5]; y = 2
14 // Expected = 0

```

Figure 3: Code 2

```

1 public static int countPositive (int[] x) {
2     // Effects: If x==null throw NullPointerException
3     // else return the number of
4     // positive elements in x.
5     int count = 0;
6     for (int i=0; i < x.length; i++) {
7         if (x[i] >= 0) {
8             count++;
9         }
10    }
11    return count;
12 }
13 // test: x=[-4, 2, 0, 2]
14 // Expected = 2

```

Figure 4: Code 3