Bài tập về nhà số 1

Lớp: INT3117_2

Sinh viên: Nguyễn Đăng Nam

Mã số sinh viên: 18020931

Câu 1. Sự *khác nhau* giữa fault và failure là gì?

Fault (Sai)	Failure (Thất bại)
Fault (sai) là *kết quả* của lỗi (error), hay nói cách khác lỗi dẫn đến sai. Sai *khó phát hiện* và là một biểu diễn của lỗi dưới dạng một biểu thức. *Ví dụ:* chương trình, văn bản, sơ đồ, biểu đồ …	Thất bại xuất hiện khi một lỗi (error) được *thực thi*. Có *hai* điều cần lưu ý về thất bại: + Thất bại *chỉ xuất hiện* ở dạng có thể thực thi mà thông thường là mã nguồn (source code). + Thất bại *chỉ liên kết* với các lỗi về nghiệp vụ

=> *Fault gây ra Failure*

Câu 2. Bài 5, trang 40, sách Introduction to Software Testing

Bài toán Find last index of element

```
/**
 * Find last index of element
 *
 * @param x array to search
 * @param y value to look for
 * @return last index of y in x; -1 if absent
 * @throws NullPointerException if x is null
 */
public int findLast (int[] x, int y)
{
  for (int i=x.length-1; i > 0; i--)
  {
    if (x[i] == y)
    {
      return i;
    }
  }
  return -1;
}
// test:  x = [2, 3, 5]; y = 2; Expected = 0
// Book website: FindLast.java
// Book website: FindLastTest.java
```

a) **Lỗi sai (fault)**: Việc chạy vòng lặp để duyệt như trên là duyệt từ phần tử cuối cùng của mảng đến phần tử thứ 2 (có vị trí index là 1) của mảng. Việc này sẽ thiếu mất việc duyệt phần tử đầu tiên, tức phần tử index 0 của mảng.

Sửa lỗi: Thay đổi vòng lặp thành:

for(int i=x.length-1; i>=0; i--)

b) *Test case không bị sai:*

x = [1, 2, 3, 4, 5]; y = 2; Expected: 1; Output: 1

Việc chạy vòng lặp này sẽ dừng tại vị trí index là 1 và vị trí cần tìm cũng tại index 1.

c) *Test case sinh lỗi:*

x = [1,2,3,4,5]; y=1; Expected: 0; Output: -1

Việc chạy vòng lặp này sai vì vị trí dừng là vị trí index 1 nhưng kết quả cần tìm ở index 0.

d) *Test case sinh lỗi nhưng không thất bại*:

x = [1,2,3,4,5]; y=6; Expected: -1; Output: -1

Việc chạy vòng lặp này tuy được kết quả đúng nhưng do không duyệt hết vòng lặp nên không thể biết được phần tử index 0 có đúng bằng y không? Thật may mắn thay là trường hợp này thì không phải.

e) Việc giải thích các lỗi với các test case ở các trường hợp đã được đề cập và giải thích ngay bên dưới của mỗi phần

Bài toán Find last index of zero

```
/**
 * Find last index of zero
 *
 * @param x array to search
 *
 * @return last index of 0 in x; -1 if absent
 * @throws NullPointerException if x is null
 */
public static int lastZero (int[] x)
{
  for (int i = 0; i < x.length; i++)
  {
    if (x[i] == 0)
    {
      return i;
    }
  }
  return -1;
}
// test:  x = [0, 1, 0]; Expected = 2
// Book website: LastZero.java
// Book website: LastZeroTest.java
```

a) **Lỗi sai (fault)**: Việc chạy vòng lặp và kết thúc vòng lặp ngay khi thấy phần tử có giá trị bằng 0 thì sai với mục đích của bài toán là find *last* index of zero

Sửa lỗi: Thay đổi vòng lặp thành:

for(int i=x.length-1; i>=0; i--)

b) *Test case không bị sai:*

x = [1,0,2,3,4]; Expected: 1; Output: 1

Việc chạy vòng lặp này sẽ dừng tại vị trí index là 1 và vị trí cần tìm cũng tại index 1.

c) *Test case sinh lỗi:*

x = [0,1,2,0]; Expected: 3; Output: 0

Việc chạy vòng lặp này sai vì vị trí dừng là vị trí index 0 nhưng kết quả cần tìm ở index 3.

d) *Test case sinh lỗi nhưng không thất bại*:

Tất cả các test case không tồn tại phần tử 0 trong mảng x đều thỏa mãn yêu cầu của đề bài

e) Việc giải thích các lỗi với các test case ở các trường hợp đã được đề cập và giải thích ngay bên dưới của mỗi phần

Bài toán Count positive elements

```
/**
 * Count positive elements
 *
 * @param x array to search
 * @return count of positive elements in x
 * @throws NullPointerException if x is null
 */
public int countPositive (int[] x)
{
  int count = 0;
  for (int i=0; i < x.length; i++)
  {
    if (x[i] >= 0)
    {
      count++;
    }
  }
  return count;
}
// test:  x = [-4, 2, 0, 2]; Expected = 2
// Book website: CountPositive.java
// Book website: CountPositiveTest.java
```

a) **Lỗi sai (fault)**: Đề bài yêu cầu đếm số lượng phần tử dương trong mảng, tuy nhiên 0 không phải là số dương nên biểu thức điều kiện *x>=0* bên trong vòng lặp là sai.

Sửa lỗi: Thay đổi biểu thức điều kiện trong vòng lặp là:

if(x[i] > 0)

b) *Test case không bị sai:*

x = [1, 2, 3, 4, 5]; Expected: 5; Output: 5

Việc mảng không có phần tử 0 khiến cho việc đếm số dương là đúng với biểu thức của đề bài.

c) *Test case sinh lỗi:*

x = [0,1,2,3,4,5]; Expected: 5; Output: 6

Việc có phần tử 0 trong mảng đã làm việc đếm số dương theo đề bài là sai và bị lệch đi so với kết quả một lượng bằng số lượng số 0 có trong mảng.

d) *Test case sinh lỗi nhưng không thất bại*:

Tất cả các test case có không chứa phần tử 0 trong mảng x đều cho ta kết quả như mong đợi (expected)

e) Việc giải thích các lỗi với các test case ở các trường hợp đã được đề cập và giải thích ngay bên dưới của mỗi phần

Bài toán Count odd or positive elements

```
/**
 * Count odd or positive elements
 *
 * @param x array to search
 * @return count of odd/positive values in x
 * @throws NullPointerException if x is null
 */
public static int odd0rPos(int[] x)
{
  int count = 0;
  for (int i = 0; i < x.length; i++)
  {
    if (x[i]%2 == 1 || x[i] > 0)
    {
      count++;
    }
  }
  return count;
}
// test:  x = [-3, -2, 0, 1, 4]; Expected = 3
// Book website: Odd0rPos.java
// Book website: Odd0rPosTest.java
```

a) **Lỗi sai (fault)**: Đề bài yêu cầu đếm phần tử lẻ hoặc số dương nhưng với biểu thức điều kiện như trên sẽ thiếu trường hợp số lẻ âm.

Sửa lỗi: Thay đổi biểu thức điều kiện thành:

if(Math.abs(x[i]%2 == 1) || x[i] > 0)

b) *Test case không bị sai:*

x = [1,2,3,4,5]; Expected: 5; Output: 5

Việc không tồn tại số lẻ âm trong mảng đã làm cho đoạn mã chạy đúng với kì vọng của bài toán

c) *Test case sinh lỗi:*

x = [1,2,3,4,5,-1]; Expected: 6; Output: 5

Việc có phần thử lẻ âm xuất hiện trong mảng x đã làm cho đoạn mã bị sai và đếm thiếu số lượng bằng chính số lượng phần tử lẻ âm

d) *Test case sinh lỗi nhưng không thất bại*:

Tất cả các test case không chứa số lẻ âm đều có output đúng với expected, tuy nhiên nó vẫn sinh ra lỗi vì không đúng với yêu cầu bài toán

e) Việc giải thích các lỗi với các test case ở các trường hợp đã được đề cập và giải thích ngay bên dưới của mỗi phần

f)

Kết quả tất cả các test case của 4 bài toán:

```
PASS test/exercise1.test.js
  √ Find last: x = [1,2,3,4,5]; y = 2 (2 ms)
  √ Find last: x = [1,2,3,4,5]; y=1
  √ Find last: x = [1,2,3,4,5]; y=0
  √ Last zero: x = [1,0,2,3,4]
  √ Last zero: x=[0,1,2,0]
  √ Last zero: x = [1,2,3,4]
  √ Count positive: x = [1,2,3,4,5)
  √ Count positive: x = [0,1,2,3,4,5)
  √ Count positive: x = [1,2,3,4,5,6) (1 ms)
  √ Odd or Pos: x = [1,2,3,4,5]
  √ Odd or Pos: x = [1,2,3,4,5,-1]
  √ Odd or Pos: x = [1,2,3,4,5,6]

Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        1.663 s
Ran all test suites.
```

Câu 3. *Mô tả* một bài toán với *đầu vào, đầu ra* và *yêu cầu chức năng rõ ràng*. Bài toán này sẽ được sử dụng để làm bài tập trong toàn khóa học. Tham khảo các ví dụ trong chương 2 sách giáo trình

Bài toán đưa ra:

Tính toán tiền điện của hộ gia đình theo số điện tiêu dùng đã dùng

TT	Nhóm đối tượng khách hàng	Gía bán điện (đồng/kWh)
1	Bậc 1: Cho kWh từ 0 - 50	1.549
2	Bậc 2: Cho kWh từ 51 - 100	1.600
3	Bậc 3: Cho kWh từ 101 - 200	1.858
4	Bậc 4: Cho kWh từ 201 - 300	2.340
5	Bậc 5: Cho kWh từ 301 - 400	2.615
6	Bậc 6: Cho kWh từ 401 trở lên	2.701

Bài toán có dữ liệu đầu vào là

+ Số kWh điện đã dùng: ***e***

Yêu cầu chức năng:

+ Từ dữ liệu đầu vào và bảng tính giá tiền để quy ra số diền mà cá nhân, đơn vị phải đóng

Ví dụ:

Input: 150

Output: 250350

Giải thích:

Số tiền điện phải đóng là:

*1.549*50 + 1.600*50 + 1.858*50 = 250350*

Nhận xét:

Vấn đề phức tạp của bài toán là chia các đoạn tiêu thụ điện năng ra với các mức giá tiền khác nhau, từ đó tính tổng chi phí các đoạn chi phí tiêu thụ điện năng để thu được kết quả cuối cùng.