ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÀI TẬP

KIỂM THỬ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

Lớp học phần: INT3117 1

Giảng viên: Nguyễn Thu Trang

Sinh viên

Họ tên: Nguyễn Thị Xuân

MSSV: 18021451

Bài tập: Chọn hai phương pháp kiểm thử hộp đen để kiểm thử bài toán của mình

Bài toán: Cho số nguyên dương n và a (a < 100, n < 100). Hỏi: nếu a là số hoàn thiện thì a và n có nguyên tố cùng nhau không?
Input: số tự nhiên n,
Output: cố các số hoàn thiện có trong mảng a thỏa mãn < n.
Mã nguồn:

```
public class CountPerfectNumber {
   public int n;
   public int a;

   public CountPerfectNumber(int n, int a) {
        this.n = n;
        this.a = a;
   }

   boolean  check(){
        if(checkPerfectNumber(n)){
            if(checkTwoNumbers(n, a)){
                return  true;
            }
        }
        return false;
   }
   public boolean checkTwoNumbers(int a, int b) {
        int gcd = 1;
        if (a == 0 || b == 0) {
            gcd = a + b;
        }
        while (a != b) {
            if (a > b) {
                a -= b;
            } else {
                b -= a;
            }
        }
        gcd = a;
        if (gcd == 1) return true;
        else return false;
   }

   public boolean checkPerfectNumber(int num) {
        int sum = 0;
        for (int i = 1; i < num; i++) {
            if (num % i == 0) {
                sum += i;
            }
        }
        if (sum == num) {
            return true;
        } else {
            return false;
        }
   }
}
```

1. Kiểm thử giá trị biên

Input :

- Số tự nhiên a thuộc [0, 100]

- Số tự nhiên n thuộc [0, 100]

Output:

-True/False

Blến	**min**	**min+**	**normal**	**max-**	**max**
a	0	1	35	99	100
n	0	1	6	99	100

Từ bảng miền giá trị trên ta sinh ra được các ca kiểm thử như sau:

Các ca kiểm thử:

(noma, nomn), (noma, minn), (noma, minn+), (noma, maxn-), (noma, maxn), (mina, nomn), (mina+, nomn), (maxa, nomn), (maxa-, nomn)

Test case	**Test Input**		**Expected Output**	**Actual Output**	**Result**
	a	**n**			
1	35	6	True	True	Pass
2	35	0	False	False	Pass
3	35	1	False	False	Pass
4	35	99	False	False	Pass
5	35	100	False	False	Pass
6	0	6	False	False	Pass
7	1	6	False	False	Pass
8	99	6	False	False	Pass
9	100	6	False	False	Pass

Mã test sử dụng Junit test

```
@Test
public void testCaseBoundary7() {
    CountPerfectNumber countPerfectNumber = new CountPerfectNumber( n: 1,  a: 6);
    boolean check = countPerfectNumber.check();
    assertEquals( expected: false, check);
}
@Test
public void testCaseBoundary8() {
    CountPerfectNumber countPerfectNumber = new CountPerfectNumber( n: 99,  a: 6);
    boolean check = countPerfectNumber.check();
    assertEquals( expected: false, check);
}
@Test
public void testCaseBoundary9() {
    CountPerfectNumber countPerfectNumber = new CountPerfectNumber( n: 100,  a: 6);
    boolean check = countPerfectNumber.check();
    assertEquals( expected: false, check);
}
```

2. Kiểm thử bảng quyết định

Bảng quyết định:

Điều kiện kiểm tra: n có phải số hoàn thiện không? nếu n là số hoàn thiện thì a và n có nguyên tố cùng nhau không?

Điều kiện	0 < a< 100	T	T	T	-	F
	0 < n < 100	T	T	T	F	-
	n là số hoàn thiện	T	T	F	-	-
	UCLN(a, n) = 1	T	F	-	-	-
Hành động	đánh dấu n là số hoàn thiện	x	x			
	UCLN(a, n) = 1	x				
	Return True	x				
	Return False		x	x	x	x

Từ bảng quyết định ta có các test case sau:

testCase	Test input		Expected Output	Actual Output	Result
	n	a			
1	6	35	True	True	Pass
2	6	36	False	False	Pass
3	7	8	False	False	Pass
4	150	2	False	False	Pass
5	28	-10	False	False	Pass

Mã test sử dụng Junit Test

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class CountTesting {
   public CountTesting(){}
   @Test
   public void testCase1() {
        CountPerfectNumber countPerfectNumber = new CountPerfectNumber(6,
35);
        boolean check = countPerfectNumber.check();
        assertEquals(true, check);
   }
   @Test
   public void testCase2() {
        CountPerfectNumber countPerfectNumber = new CountPerfectNumber(6,
36);
        boolean check = countPerfectNumber.check();
        assertEquals(false, check);
   }
   @Test
   public void testCase3() {
        CountPerfectNumber countPerfectNumber = new CountPerfectNumber(7,
8);
        boolean check = countPerfectNumber.check();
        assertEquals(false, check);
   }
   @Test
   public void testCase4() {
        CountPerfectNumber countPerfectNumber = new
CountPerfectNumber(150, 2);
        boolean check = countPerfectNumber.check();
        assertEquals(false, check);
   }
   @Test
   public void testCase5() {
        CountPerfectNumber countPerfectNumber = new CountPerfectNumber(28,
-10);
        boolean check = countPerfectNumber.check();
```

```
        assertEquals(false, check);
    }
}
```