**Lab#7 – White-box testing**

# วัตถุประสงค์การเรียนรู้

1. ผู้เรียนสามารถออกแบบการทดสอบแบบ White-box testing ได้
2. ผู้เรียนสามารถวิเคราะห์ปัญหาด้วย Control flow graph ได้
3. ผู้เรียนสามารถออกแบบกรณีทดสอบโดยคำนึงถึง Line coverage ได้
4. ผู้เรียนสามารถออกแบบกรณีทดสอบโดยคำนึงถึง Block coverage ได้
5. ผู้เรียนสามารถออกแบบกรณีทดสอบโดยคำนึงถึง Branch coverage ได้
6. ผู้เรียนสามารถออกแบบกรณีทดสอบโดยคำนึงถึง Condition coverage ได้
7. ผู้เรียนสามารถออกแบบกรณีทดสอบโดยคำนึงถึง Branch and Condition coverage ได้

# โจทย์: Clump counts

Clump counts (<https://codingbat.com/prob/p193817>) เป็นโปรแกรมที่ใช้ในการนับการเกาะกลุ่มกันของข้อมูลภายใน Array โดยการเกาะกลุ่มกันจะนับสมาชิกใน Array ที่อยู่ติดกันและมีค่าเดียวกันตั้งแต่สองตัวขึ้นไปเป็นหนึ่งกลุ่ม เช่น

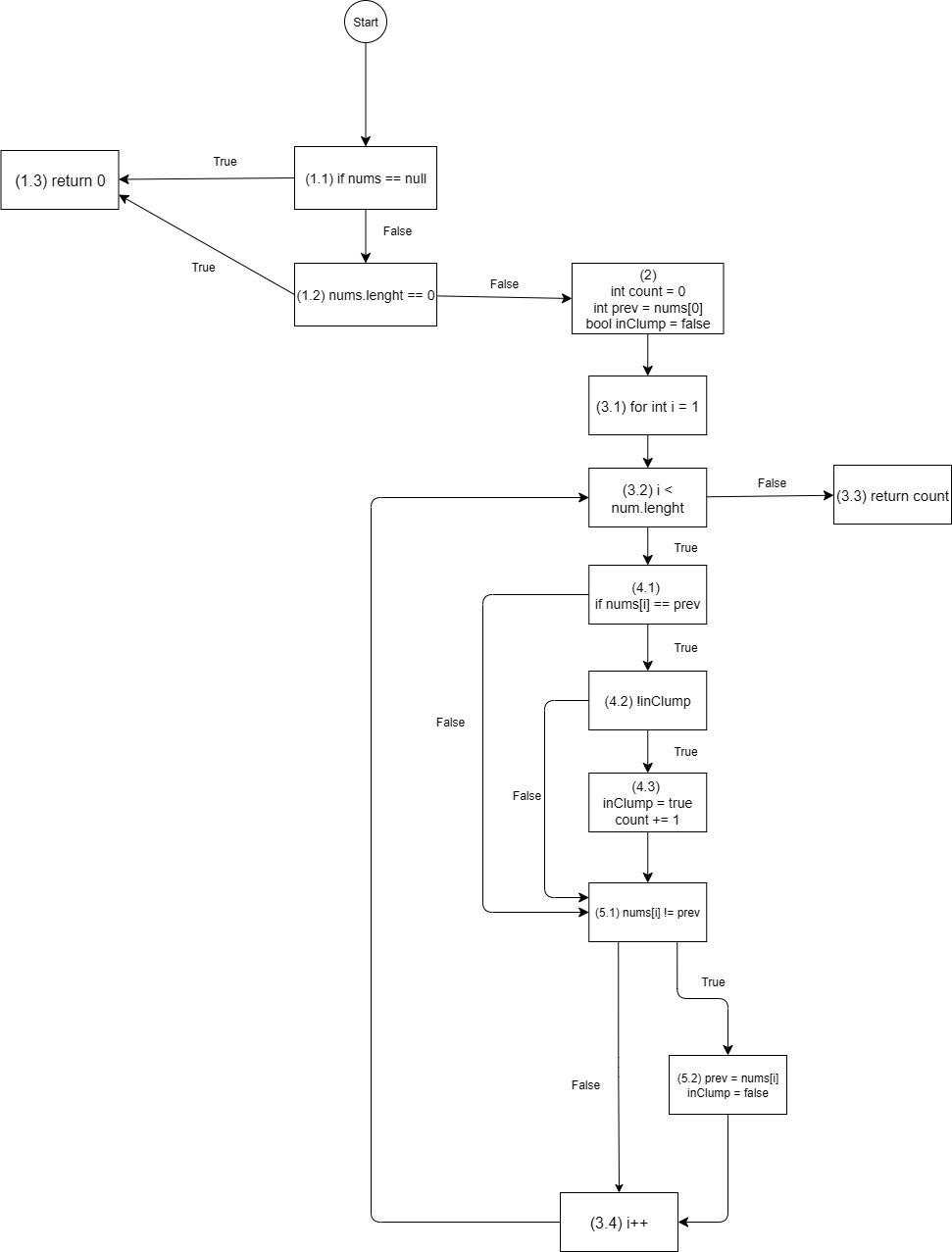
[1, 2, 2, 3, 4, 4] → 2

[1, 1, 2, 1, 1] → 2

[1, 1, 1, 1, 1] → 1

ซอร์สโค้ดที่เขียนขึ้นเพื่อนับจำนวนกลุ่มของข้อมูลที่เกาะอยู่ด้วยกันอยู่ที่ <https://github.com/ChitsuthaCSKKU/SQA/tree/2025/Assignment/Lab7> โดยที่ nums เป็น Array ที่ใช้ในการสนับสนุนการนับกลุ่มของข้อมูล (Clump) ทำให้ nums เป็น Array ที่จะต้องไม่มีค่าเป็น Null และมีความยาวมากกว่า 0 เสมอ หาก nums ไม่เป็นไปตามเงื่อนไขที่กำหนดนี้ โปรแกรมจะ return ค่า 0 แทนการ return จำนวนกลุ่มของข้อมูล

# แบบฝึกปฏิบัติที่ 7.1 Control flow graph

จากโจทย์และ Source code ที่กำหนดให้ (CountWordClumps.java) ให้เขียน Control Flow Graph (CFG) ของเมธอด countClumps() จากนั้นให้ระบุ Branch และ Condition ทั้งหมดที่พบใน CFG ให้ครบถ้วน

ตอบ

Branch:

A: 1.1 = true, 1.2 = false (nums is null)

B: 1.1 = false, 1.2 = true (nums.length = 0)

C: 1.1 = false, 1.2 = false (nums is valid input)

D: 4.1 = true, 4.2 = false (prev == nums[i] & inClump = true -> ยังเจอคำเดิม)

E: 4.1 = false, 4.2 = false (prev != nums[i] & inClump = true -> เจอค่าใหม่)

F: 4.1 = true, 4.2 = true (prev == nums[i] & inClump = false -> เจอ Clump word)

G: 4.1 = false 4.2 = true (ไม่เจอ Clump word)

H: 5.1 = true (prev != nums[i] -> เปลี่ยนค่า prev = nums[i])

I: 5.1 = false (เจอคำซ้ำ)

J: 3.2 = true

K: 3.2 = false

Condition:

1.1: nums == null

1.2: nums.length == 0

3.2: i < nums.length

4.1: nums[i] == prev

4.2: !inClumps

5.1: nums[i] != prev

# แบบฝึกปฏิบัติที่ 7.2 Line Coverage

1. จาก Control Flow Graph (CFG) ของเมธอด countClumps() ในข้อที่ 1 ให้ออกแบบกรณีทดสอบเพื่อให้ได้ Line coverage = 100%
2. เขียนกรณีทดสอบที่ได้ พร้อมระบุบรรทัดที่ถูกตรวจสอบทั้งหมด
3. แสดงวิธีการคำนวณค่า Line coverage

ตอบ

| Test Case No. | Input(s) | Expected Result(s) | Path and Branch |
| --- | --- | --- | --- |
| TC01 | [] | 0 | Line No.: 5-6 |
| TC02 | [1,1,1,2,2] | 2 | Line No.: 5-25 |

Total lines = 20, Execution line = 25-5 = 20

Line coverage = (Execution Line / Total lines) \* 100

= (20 / 20) \* 100

= 1 \* 100

= 100 %

# แบบฝึกปฏิบัติที่ 7.3 Block Coverage

1. จาก Control Flow Graph (CFG) ของเมธอด countClumps() ในข้อที่ 1 ให้ออกแบบกรณีทดสอบเพื่อให้ได้ Block coverage = 100%
2. เขียนกรณีทดสอบที่ได้ พร้อมระบุ Block ที่ถูกตรวจสอบทั้งหมด
3. แสดงวิธีการคำนวณค่า Block coverage

ตอบ

| Test Case No. | Input(s) | Expected Result(s) | Path and Branch |
| --- | --- | --- | --- |
| TC01 | [] | 0 | Block:1.1, 1.2, 1.3 |
| TC02 | [1,1,1,2,2] | 2 | Block: 1.1, 1.2, 2, 3.1, 3.2, 4.1, 4.2, 4.3, 5.1, 5.2, 3.4, 3.3 |
| TC03 | Null | 0 | Block: 1.1, 1.3 |

Total Blocks = 13, Execution Blocks = 13

Block coverage = (Execution Blocks / Total Blocks) \* 100

= (13 / 13) \* 100 = 100%

# แบบฝึกปฏิบัติที่ 7.3 Branch Coverage

1. จาก Control Flow Graph (CFG) ของเมธอด countClumps() ในข้อที่ 1 ให้ออกแบบกรณีทดสอบเพื่อให้ได้ Branch coverage = 100%
2. เขียนกรณีทดสอบที่ได้ พร้อมระบุ Path และ Branch ที่ถูกตรวจสอบทั้งหมด
3. แสดงวิธีการคำนวณค่า Branch coverage

ตอบ

| Test Case No. | Input(s) | Expected Result(s) | Path and Branch |
| --- | --- | --- | --- |
| TC01 | [] | 0 | Path: 1.1 -> 1.2 -> 1.3  Branch: B |
| TC02 | [1,1,1,2,2] | 1 | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 4.2 -> 4.3 -> 5.1 -> 5.2-> 3.4 -> 3.2-> 3.3  Branch:  C (not null && nums.length > 0)  D (when prev != nums[i] but inClumps = true)  E (when prev == nums[i] but inClumps = true)  F ( when prev == nums[i] and inClumps = false  H (when prev != nums[i])  I (when prev = nums[i])  J (when loop start)  K (when loop end) |
| TC03 | Null | 0 | Path: 1.1 -> 1.3  Branch: A (when nums is null) |
| TC04 | [1, 2, 3, 4] | 0 | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 5.1 -> 5.2 -> 3.4 -> 3.2 -> 3.3  Branch:  C (not null && nums.length > 0)  G (when prev != nums[i] and inClumps = false)  H (when prev != nums[i])  J (when loop start)  K (when loop end) |

Total Branch = 11, Executed Branch = 11

Branch coverage = (Executed Branch / Total Branch) \* 100

= (11 / 11) \* 100 = 1 \* 100 = 100%

# แบบฝึกปฏิบัติที่ 7.4 Condition Coverage

1. จาก Control Flow Graph (CFG) ของเมธอด countClumps() ในข้อที่ 1 ให้ออกแบบกรณีทดสอบเพื่อให้ได้ Condition coverage = 100%
2. เขียนกรณีทดสอบที่ได้ พร้อมระบุ Path และ Condition ที่ถูกตรวจสอบทั้งหมด เช่น Condition A = T และ Condition B = F
3. แสดงวิธีการคำนวณค่า Condition coverage

ตอบ

| Test Case No. | Input(s) | Expected Result(s) | Path and Condition |
| --- | --- | --- | --- |
| TC01 | [] | 0 | Path: 1.1 -> 1.2 -> 1.3  Condition:  1.1 = false (not null)  1.2 = true (nums.length = 0) |
| TC02 | [1,1,1,2,2] | 1 | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 4.2 -> 4.3 -> 5.1 -> 3.2 -> 3.3  Condition:  1.1 = false  1.2 = false  3.2 = true (วนลูป), false(จบลูป)  4.1 = true(เจอค่าที่เท่ากับ prev), false (เจอค่าที่ไม่เท่ากับ prev)  4.2 = true (ไม่ได้เจอค่าซ้ำ), false (เจอค่าซ้ำ)  5.1 = true(จอค่าที่ไม่เท่ากับ prev),false(เจอค่าที่เท่ากับ prev) |
| TC03 | Null | 0 | Path: 1.1 -> 1.3  Condition:  1.1 = true (nums = null) |
| TC04 | [1, 2, 3, 4] | 0 | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 5.1 -> 5.2 -> 3.4 -> 3.2 -> 3.3  Condition:  1.1 = false  1.2 = false  3.2 = true (วนลูป), false(จบลูป)  4.1 = false (เจอค่าที่ไม่เท่ากับ prev)  4.2 = true (ไม่ได้เจอค่าซ้ำจึงเป็น)  5.1 = true (prev != nums[i]) |

Total Condition = 12, Executed Condition = 12

Condition coverage = (Executed Condition / Total Condition) \* 100

= (12 / 12) \* 100 = 1 \* 100 = 100%

# แบบฝึกปฏิบัติที่ 7.5 Branch and Condition Coverage (C/DC coverage)

1. จาก Control Flow Graph (CFG) ของเมธอด countClumps() ในข้อที่ 1 ให้ออกแบบกรณีทดสอบให้ได้ C/DC coverage = 100%
2. เขียนกรณีทดสอบที่ได้ พร้อมระบุ Path, Branch, และ Condition ที่ถูกตรวจสอบทั้งหมด
3. แสดงวิธีการคำนวณค่า C/DC coverage
4. เขียนโค้ดสำหรับทดสอบตามกรณีทดสอบที่ออกแบบไว้ด้วย JUnit และบันทึกผลการทดสอบ

ตอบ

| Test Case No. | Input(s) | Expected Result(s) | Actual Result(s) | Path, Branch, and Condition |
| --- | --- | --- | --- | --- |
| TC01 | [] | 0 | 0  Pass/Fail: Pass | Path: 1.1 -> 1.2 -> 1.3  Branch: B  Condition:  1.1 = false (not null)  1.2 = true (nums.length = 0) |
| TC02 | [1,1,1,2,2] | 2 | 2  Pass/Fail: Pass | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 4.2 -> 4.3 -> 5.1 -> 3.2 -> 3.3  Branch:  C (not null && nums.length > 0)  D (when prev != nums[i] but inClumps = true)  E (when prev == nums[i] but inClumps = true)  F ( when prev == nums[i] and inClumps = false  H (when prev != nums[i])  I (when prev = nums[i])  J (when loop start)  K (when loop end)  Condition:  1.1 = false  1.2 = false  3.2 = true (วนลูป), false(จบลูป)  4.1 = true(เจอค่าที่เท่ากับ prev), false (เจอค่าที่ไม่เท่ากับ prev)  4.2 = true (ไม่ได้เจอค่าซ้ำ), false (เจอค่าซ้ำ)  5.1 = true(จอค่าที่ไม่เท่ากับ prev),false(เจอค่าที่เท่ากับ prev) |
| TC03 | null | 0 | Pass  Pass/Fail: Pass | Path: 1.1 -> 1.3  Branch: A (when nums is null)  Condition:  1.1 = true (nums = null) |
| TC04 | 1,2,3,4 | 0 | Pass  Pass/Fail: Pass | Path: 1.1 -> 1.2 -> 2 -> 3.1 -> 3.2 -> 4.1 -> 5.1 -> 5.2 -> 3.4 -> 3.2 -> 3.3  Branch:  C (not null && nums.length > 0)  G (when prev != nums[i] and inClumps = false)  H (when prev != nums[i])  Condition:  1.1 = false  1.2 = false  3.2 = true (วนลูป), false(จบลูป)  4.1 = false (เจอค่าที่ไม่เท่ากับ prev)  4.2 = true (ไม่ได้เจอค่าซ้ำจึงเป็น)  5.1 = true (prev != nums[i]) |

Total Branch and Condition = 11+12 = 23

Execution Branch and Condition = 11+12 = 23

C/DC coverage = (Execution Branch and Condition / Total Branch and Condition) \* 100

= (23/23) \* 100 = 100%