CS7314 Homework #4

Name: Bingying Liang
ID: 48999397

Due: Mar 27 2024

1. Problem 11.1 or 11.2 (pick one) and the following extensions:

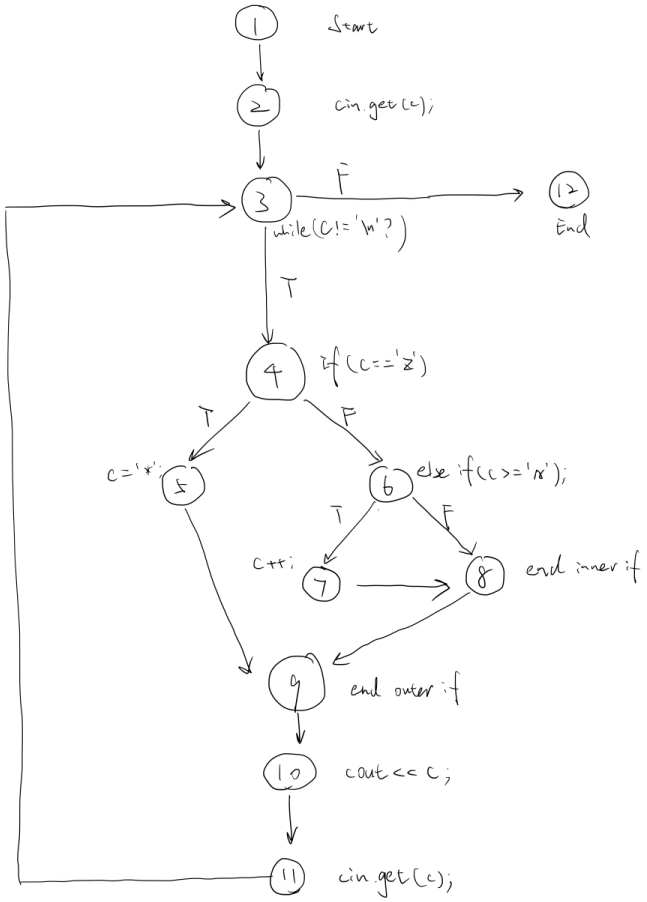
   - Determine the number of paths and sensitize a few of your test cases.
   - If your CFG contains no loops, try to modify it to include at least one loop.
   - What is your loop testing strategy, and the corresponding test cases for it?
   - What is your loop testing strategy when you introduce nested loops in your CFG?

   Solution:

   I choose 11.1 : Construct a CFG for a small program. The details of the code is in the following:

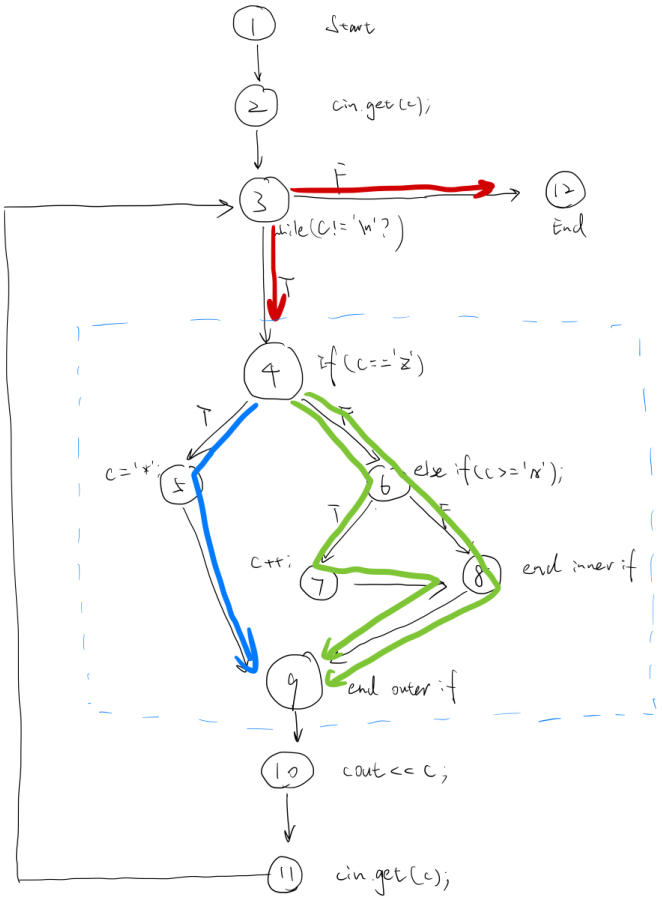
```
cin.get(c);
while (c != '\n'){
  if (c == 'z'){
    c = '*';
  }else if (c >= 'A'){
    c++;
  }
  cout << c;
  cin.get(c);
}
```

   The CFG is:



(a) The number of paths: 4

$(1, 2, 3, 4, 5, 9, 10, 11, 3, 12)$,
$(1, 2, 1, 2)$,
$(1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 3, 12)$,
$(1, 2, 3, 4, 6, 8, 10, 11, 3, 12)$



The graph is nesting: G = G1 (G2), such as node 3 has 2 paths which is in the G1; Node 4, 5, 6, 7, 8, 9 which can generate 3 paths which is in G2. Therefore, $M + N - 1 = 2 + 3 - 1 = 4$ paths.

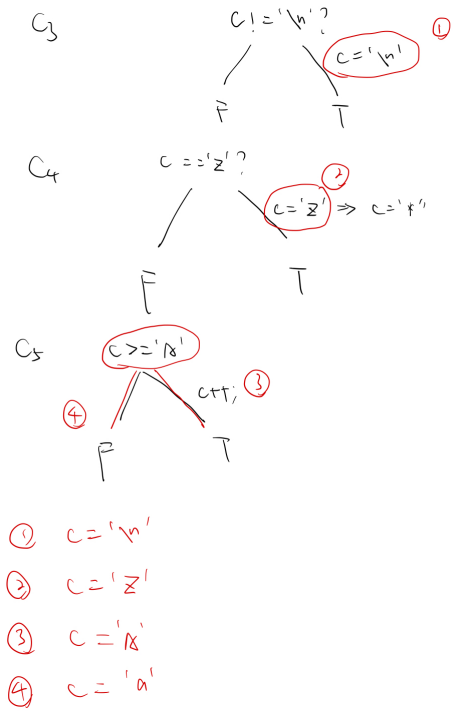
Sensitizing a few of test cases:

T1 : c = '/ n'

T2 : c = 'z'

T3 : c = 'A'

T4 : c = 'a'



(b) The program already contains an input-based loop.

(c) Loop Testing Strategy: For the original single loop, the testing strategy includes ensuring all branches are tested: Input a sequence of characters including 'z', characters 'A' to 'Z', and other characters, ending with '/ n'.

Corresponding Test Cases: Test cases include Only inputting '/ n' (to test immediate termination of the loop). Inputting a string containing 'z', 'A', and other characters, followed by '/ n'.

(d) If nested loops are introduced, the testing strategy needs to account for all possible interactions between the inner and outer loops:

- Ensure the inner loop executes correctly in each iteration of the outer loop.
- Test that the inner loop can terminate correctly without affecting the continuation of the outer loop.
- Ensure that when the outer loop's termination condition is met, the outer loop can terminate correctly regardless of the state of the inner loop.

For nested loops, test cases could be designed as follows:

- Input a sequence of characters that causes the inner loop to execute a specific number of times before satisfying the outer loop's termination condition.
- Input a sequence of characters that never satisfies the inner loop's termination condition (if the logic allows) to observe the outer loop's response.

2. Problem 11.4e and 11.4f.

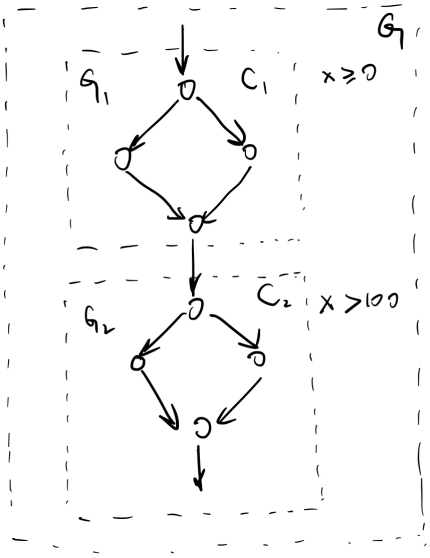
Consider the sensitization of a CFG that consists of sequential concatenation of two subgraphs G1 and G2 (that is, G = G1 o G2). G1 and G2 each contains a binary branching with conditions C1 and C2 respectively. Discuss the possible sensitization issues for the following cases:

e) C2 $\Rightarrow$ C1.

f) C1 and C2 overlaps, for example, C1 = $(0 \leq x \leq 100)$ and C2 = $(50 \leq x \leq 200)$

Solution:

e) During the problem information, I can draw the CFG in the following:



Suppose C2: $x > 100$, C1: $x \geq 0$.

When C2 implies C1, it means that once C2 is true, then C1 must also be true. This introduces some complexity in the sensitization of CFG, as the path selection in G2 is directly affected by the conditional judgment results in G1. In this case, if the path that makes C1 false is selected in G1, then the path that makes C2 true cannot be selected in G2. This requires that the implication relationship between C2 and C1 be taken into account in path selection and test case design to ensure that the test covers all valid path combinations, especially if C2 is true and C1 must also be true.

Therefore, Sensitization Test Cases can be in the following:

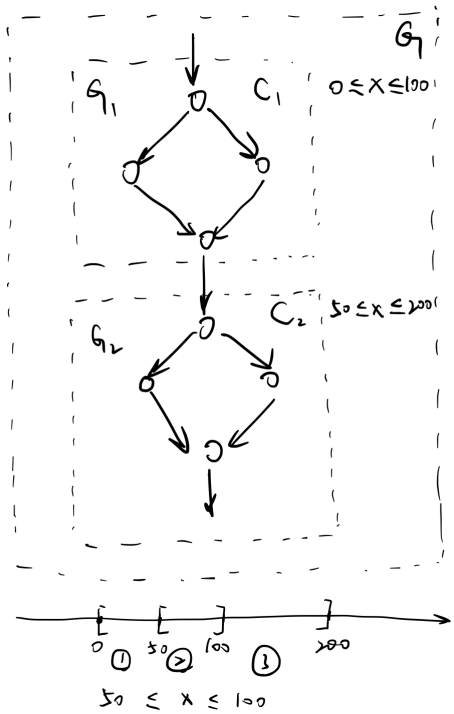
C_1	C_2	Test cases for x
T	F	x = 100
T	T	x = 200
F	T	—
F	F	x = -1

T1 : x = 100;

T2 : x = 200;

T3 : x = -1;

f) During the problem information, I can draw the CFG in the following:



When C1 and C2 overlap, for example C1 defines x in the range 0 to 100, and C2 defines x in the range 50 to 200, this means that there is an interval of x values (50 to 100) in which both C1 and C2 will be true. The problem with sensitization in this case is the need to ensure that the test case covers the intersection of C1 and C2, as well as the cases where only C1 is true and only C2 is true. This requires test cases to consider not only the overlap of the two conditions, but also the parts that are unique to each, ensuring that all possible paths in CFG are adequately tested.

Therefore, Sensitization Test Cases can be in the following:

C_1	C_2	Test cases for x
T	T	x = 75
T	F	x = 25
F	T	x = 150
F	F	x = -1

T1 : x = 75;
T2 : x = 25;
T3 : x = 150;
T4 : x = -1;

3. Problem 11.9 or 11.10 (pick one).

   (Since we have just started this topic, a simple program/function/segment would be OK.)

Solution:
I choose Problem 11.9: Construct a DDG for a small program. The details of the code is in the following:

```
public class SimpleCalculation {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        int sum = a + b;
        int product = a * b;
        int difference = b - a;
        int result = sum + product - difference;
    }
}

```