Darshan Pradhan

SE 333: Software Testing

5/10/2021

**Assignment 6: Test coverage Design**

1.

Given the following method in Java...

/\*\*

\* Returns the subset of ints that are greater than the provided

\* target value

\* @param a: an array of ints to search

\* @param target: the value to search for

\* @return the subset

\*/

public static int [] greaterThan (int [] a, int target) {

List<Integer> found = new ArrayList<>();

if (a != null && a.length > 0) {

for (int i=0; i < a.length; i++) {

if (a[i] > target) {

found.add(a[i]);

}

}

return found.stream().mapToInt(Integer::intValue).toArray();

}

throw new IllegalArgumentException("Input array is null or empty");

}

1. Design a test suite with the fewest number of test cases that satisfies the statement test criterion. Define the test suite as one or more literal arrays:

T1 = [x, y, …], n

Notice that a test case requires 2 values:

1. an array to search

2. an int to search for

T1 = [], 2

T2 = [2, 3, 4], 3

T3 = [2, 3, 4, 5], 4

T4 = [4], 1

1. Design a test suite with the fewest number of test cases that satisfies the branch-condition test criterion. Define this test suite using the same technique as above

T1 = [], 2

T2 = [2,3,4,5], 2

2.

Consider the following code fragment in C:

int max\_size = 10;

int count = 0;

for (n = 0; n < max\_size && (c = getc(yyin))!= EOF’; ++n) {

count++;

if (count < 10) {...}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | A | B | C |  |
| ID | input | n < max\_size | c = getc(yyin  != EOF | count < 10 | Result |
| 1 | n = 5;  c = ‘X’;  count=7 | True | True | True | True |
| 2 | n = 6;  c = ‘Y’;  count=11 | True | True | False | False |
| 3 | n = 7;  c = EOF;  count=6 | True | False | True | False |
| 4 | n = 8;  c = EOF;  count=12 | True | False | False | False |
| 5 | n = 11;  c = ‘Z’;  count=5 | False | True | True | False |
| 6 | n = 12;  c = ‘A’;  count=13 | False | True | False | False |
| 7 | n = 13;  c = EOF;  count=8 | False | False | True | False |
| 8 | n = 14;  c = EOF;  count=14 | False | False | False | False |

1. Derive a subset of test cases from the table above that satisfy the modified condition (MC/DC) adequacy criterion with respect to the loop. Express these test cases using their IDs and the clause that they test:

A = 1 and 5 (means test case 1 and test case 5 combine to satisfy MC/DC test

adequacy for condition A)

B = 1 and 3 (means test case 1 and test case 3 combine to satisfy MC/DC test

adequacy for condition B)

C = 1 and 2 (means test case 1 and test case 2 combine to satisfy MC/DC test

adequacy for condition C)

3.

Consider the following loop statement in C:

Int max\_size = 10;

for (n = 0; n < max\_size && (c = getc(yyin))!= EOF && c != ‘\n’; ++n)

buf[n] = (char) c;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | A | B | C |  |
| ID | input | n <max\_size | c = getc(yyin)  != EOF | C != ‘\n’ | Result |
| 1 | n = 5; c =‘X’ | True | True | True | True |
| 2 | n = 6; c =‘\n’ | True | True | False | False |
| 3 | n = 7; c = EOF | True | False | True | False |
| 4 | n = 8; c =‘\n’EOF | True | False | False | False |
| 5 | n = 11; c =‘Z’ | False | True | True | False |
| 6 | n = 12; c =‘\n’ | False | True | False | False |
| 7 | n = 13; c = EOF | False | False | True | False |
| 8 | n = 14; c = ‘\n’EOF | False | False | False | False |

1. Notice this problem is very similar but not identical to problem 3.
2. Derive a set of test cases that satisfy the compound condition adequacy criterion with respect to the loop. Document your test cases by extending the table above. Do not provide any unrealistic test cases
3. Calculate the MC/DC subset as in problem 2. How is this subset different from the set of test cases defined in 3b above. You do not need to actually display the “A = 1 and 3 …” list if you don’t want to, just answer the question. If showing the pairs list helps you express your answer then by all means, do show it.

A = 1 and 5 (means test case 1 and test case 5 combine to satisfy MC/DC test

adequacy for condition A)

B = 1 and 3 (means test case 1 and test case 3 combine to satisfy MC/DC test

adequacy for condition B)

C = 1 and 2 (means test case 1 and test case 2 combine to satisfy MC/DC test

adequacy for condition C)

4. Let us consider the following if statement in Java

// assume parseArray.length is 4

if ( pos < parseArray.length &&

(parseArray[pos] == ‘{‘ ||

parseArray[pos] == ‘}‘ ||

parseArray[pos] == ‘|‘) ) {

continue;

}

Derive a set of test cases and show that it satisfies the modified condition (MC/DC) adequacy criterion for this statement. For brevity, abbreviate each of the basic condition as follows:

Room pos < parseArray.length

Open parseArray[pos] == ‘{‘

Close parseArray[pos] == ‘}‘

Bar parseArray[pos] == ‘|‘

Provide the answer in 2 parts:

1. a table showing the test cases you have designed, similar to this one:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | input | Room | Open | Close | Bar | Result |
| 1 | pos=2;  pa[pos]=’{’ | True | True | True | True | True |
| 2 | pos=3;  pa[pos]=’{’ | True | True | True | False | True |
| 3 | pos=1;  pa[pos]=’{’ | True | True | False | True |  |
| 4 | pos=2;  pa[pos]=’{’ | True | True | False | False |  |
| 5 | pos=3;  pa[pos]=’|’ | True | False | True | True |  |
| 6 | pos=2;  pa[pos]=’}’ | True | False | True | False |  |
| 7 | pos=3;  pa[pos]=’|’ | True | False | False | True |  |
| 8 | pos=1;  pa[pos]=’b’ | True | False | False | False |  |
| 9 | pos=5;  pa[pos]=’}’ | False | True | True | True |  |
| 16 | pos=5;  pa[pos]=’a’ | False | False | False | False | False |

2. List of pairs of test cases similar to what you did in question 2:

Room = 1 and 9

Open = 1 and 5

Close = 6 and 8

Bar = 7 and 8