CS 3250: Software Testing (Spring 2026)

POTD 7: Logic coverage for source code (MuiltipleOccur)

Due 26-March-2026, 11:00am EST

Purpose: Practice applying logic-based testing for program source code; get ready to work on homework assignment, and prepare for quiz 4 and the final exam

You may make a copy of a worksheet and complete this activity, or type your answers in any text editor. You may work alone or with another student in this course.

Consider the following Java method, `MultipleOccur()`, and tests t1 - t8

```
Line  1:  public static String MultipleOccur(String str, Character key)
Line  2:  {
Line  3:     if (str == null || key == null)
Line  4:        throw new IllegalArgumentException("can't pass in null");  // IAE
Line  5:
Line  6:     int occurrences = 0;
Line  7:     for (int i = 0; i < str.length(); i++)
Line  8:     {
Line  9:        var currChar = str.charAt(i);
Line 10:
Line 11:        if (key.equals(currChar))
Line 12:           occurrences += 1;
Line 13:     }
Line 14:
Line 15:     if (occurrences > 1)
Line 16:        return "yes";
Line 17:     else
Line 18:        return "no";
Line 19:  }
```

The given tests (t1 - t8):

```
t1:  str = "occurrence",  key = 'c'
t2:  str = "",            key = 'c'
t3:  str = "occurrence",  key = 'r'
t4:  str = "occurrence",  key = 'z'
t5:  str = null,          key = null
t6:  str = null,          key = 'y'
t7:  str = "occurrence",  key = null
t8:  str = "occurrence",  key = 'n'
```

Identify a **minimal** set of tests for MultipleOccur() that achieves **Correlated Active Clause Coverage (CACC)**.

You **must** use tests from the list of eight tests (t1, t2, …, t8) given above, for credit.

Show your analysis and thought process.

[optional] You may start by filling out the following table. The first row is filled for you. This table will not be graded but it may be helpful when you answer the questions below.

Due to the space limitation, let's label the following with letters

```
str == null                 - we will call it "a" (column a, below)
key == null                 - we will call it "b" (column b, below)
i < str.length()            - we will call it "c" (column c, below)
key.equals(currChar)        - we will call it "d" (column d, below)
occurrences > 1             - we will call it "e" (column e, below)
```

[optional] You may start by filling out the following table. The values for test t1 are already filled in for you. (F = False, T = True, n/a = not applicable).

test#	str	key	MultipleOccur(str,key)	a	b	c	d	e
t1	“occurrence”	‘c’	Yes	F	F	T	T	F
t2	“”	‘c’						
t3	“occurrence”	‘r’						
t4	“occurrence”	‘z’						
t5	null	null						
t6	null	‘y’						
t7	“occurrence”	null						
t8	“occurrence”	‘n’						

Sample solution

test	str	key	MultipleOccur(str,key)	a	b	c	d	e
t1	“occurrence”	‘c’	“yes”	F	F	T	T	T
t2	“”	‘c’	**“no”**	**F**	**F**	**F**	**n/a**	**F**
t3	“occurrence”	‘r’	**“yes”**	**F**	**F**	**T**	**T**	**T**
t4	“occurrence”	‘z’	**“no”**	**F**	**F**	**T**	**F**	**F**
t5	null	null	**IAE**	**T**	**T**	**n/a**	**n/a**	**n/a**
t6	null	‘y’	**IAE**	**T**	**F**	**n/a**	**n/a**	**n/a**
t7	“occurrence”	null	**IAE**	**F**	**T**	**n/a**	**n/a**	**n/a**
t8	“occurrence”	‘n’	**“no”**	**F**	**F**	**T**	**T**	**F**

Since the given source code consists of 4 predicates, **CACC** tests must be created from all four predicates. (note: **all** predicates of the program under test must be considered when creating tests from source code).

For P1 = (str == null || key == null): there are two clauses, which are (a denotes `str == null`) and (b denotes `key == null`).

To determine tests for P1:

- First, plug in the truth table (exactly as we'd normally do, given a predicate with 2 clauses)
- Determine pairs of rows that satisfy CACC
- Then, map pairs of rows to the given set of tests (t1, …, t8)

row	a	b	P1 = (a ∨ b)	Pa	Pb
1	T	T	T		
2	T		T	T	
3		T	T		T
4				T	T

blank represents F

From the above truth table, CACC-adequate tests for P1:

- clause a: (row2, row4) —> (TF, FF)
- clause b: (row3, row4) —> (FT, FF)

Consider the possible pairs of rows from this truth table vs. the given tests (t1, …, t8) for the truth values written in `abcde` format. (where x means don't know or don't care yet)

- TFxxx: possible test is t6
- FTxxx: possible test is t7
- FFxxx: possible tests are t1, t2, t3, t4, t8

For P2 = (i < str.length()): there is only one clause, which is (c denotes `i<str.length()`)

Plug in the truth table

row	c	P2 = (c)	Pc
1	T	T	T
2			T

blank represents F

Possible pairs of rows that satisfy CACC for P2: (row1, row2) → (T, F).

To reach P2, P1 must be reached and P1 must be F. For P1 to be F, both clauses a and b must be F. Putting the truth values of clauses a, b, and c together: tests that satisfy CACC for P2 are FFT and FFF.

Consider the possible pairs of rows from this truth table vs. the given tests (t1, …, t8) for the truth values written in `abcde` format.

- FFTxx: possible tests are t1, t3, t4, t8
- FFFxx: possible test is t2

For P3 = (key.equas(currChar)): there is only one clause, which is (d denotes `key.equals(currChar)`)

Plug in the truth table

row	d	P3 = (d)	Pd
1	T	T	T
2			T

blank represents F

Possible pairs of rows that satisfy CACC for P3: (row1, row2) → (T, F).

To reach P3, P2 must be reached and P2 must be T. Therefore, a=F, b=F, and c=T. Putting the truth values of clauses a, b, c, and d together: tests that satisfy CACC for P3 are FFTT and FFTF.

Consider the possible pairs of rows from this truth table vs. the given tests (t1, …, t8) for the truth values written in `abcde` format.

- FFTTx: possible tests are t1, t3, t8
- FFTFx: possible test is t4

For P4 = (occurrences > 1): there is only one clause, which is (e denotes `occurrences > 1`)

Plug in the truth table

row	e	P4 = (e)	Pe
1	T	T	T
2			T

blank represents F

Possible pairs of rows that satisfy CACC for P4: (row1, row2) → (T, F).

To reach P4, P1 must be reached and P1 must be F. Therefore, a=F, b=F. Putting the truth values of clauses a, b, c, d, and e together: tests that satisfy CACC for P4 are FFxxT and FFxxF.

- For e to be T, c must be T (not empty string) and d must be T (key is found in str)

Consider the possible pairs of rows from this truth table vs. the given tests (t1, …, t8) for the truth values written in `abcde` format.

- FFTTT: possible tests are t1, t3
- FFxxF: possible test is t2, t4, t8

Combine tests chosen for P1, P2, P3, and P4

P1 requires

- TFxxx: possible test is **t6**
- FTxxx: possible test is **t7**
- FFxxx: possible tests are t1, t2, t3, t4, t8).
  - t2 and t4 are already required by P2 and P3; either t1 or t3 will be picked for P4.
  - Thus, (either t2 or t4) or (either t1 or t3) can be used for this case. Do not pick another test.

P2 requires

- FFTxx: possible tests are t1, t3, t4, t8
  - t4 is already required by P3; either t1 or t3 will be picked for P4.
  - This case can use t4, or one of (t1 or t3). Do not pick another test.
- FFFxx: possible test is **t2**

P3 requires

- FFTTx: possible tests are t1, t3, t8
  - Either t1 or t3 will be picked for P4.
  - This case can use one of (t1 or t3). Do not pick another test.
- FFTFx: possible test is **t4**

P4 requires

- FFTTT: possible tests are t1, t3
  - Need one of (t1 or t3), not both (otherwise, not minimal)
- FFxxF: possible test is t2, t4, t8
  - t2 and t4 are already required by P2 and P3.
  - Thus, either t2 or t4 can be used for this case. Do not pick another test.

A test set must have t6, t7, t2, t4, one of (t1 or t3)

Possible answers are {t1, t2, t4, t6, t7} or **{t2, t3, t4, t6, t7}** — Do not pick both t1 and t3, must choose one. Otherwise, not a minimal test set.

Grading rubric

[Total: 10 points]: Done (or provide evidence of your attempt, full or reasonable effort)

- (5 points) — Providing evidence of your attempt, minimal effort

(-2.5 points) for 24 hours late (submitted after 26-Mar-2026 11am EST, by 27-Mar-2026 11am EST)
(-5 points) for 48 hours late (submitted after 27-Mar-2026 11am EST, by 28-Mar-2026 11am EST)

Submission

- Take a selfie (or picture) of your team — get creative and have fun! — and submit it with your POTD.
- You may do one of the following:
  - take picture(s) of your POTD — if you write your answer(s) or draw a graph on papers, or
  - save your POTD as a .pdf file — No Word document.
- Upload your report (.pdf) to ***POTD 7 on Gradescope***.
- Make sure you connect your partner to your group on Gradescope so that everyone receives credit
- Each team submits only ***one*** copy

Making your submission available to instructor and course staff is ***your*** responsibility; if we cannot access or open your file, you will not get credit. Be sure to test access to your file before the due date.

Last updated 2026-03-22 16:41