Knowledge Test K8

COMP 4081 • Software Engineering • Fall 2019

Solutions

Name: _____, _____
Last name First name

Rules:

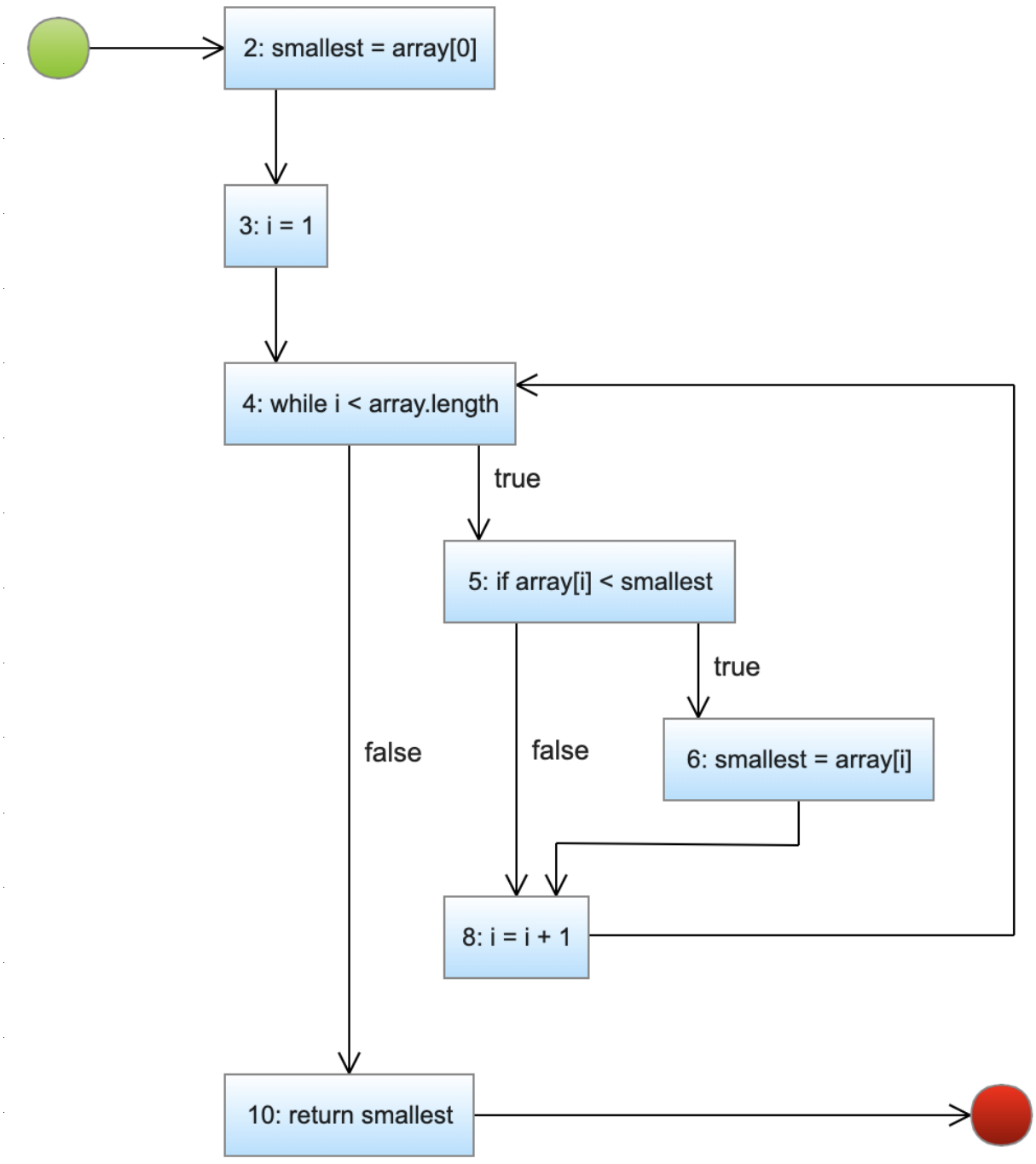
- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

- WEIRD! Do not write on the backs of pages. If you need more pages, ask me for some.

Reminders:

- Verify that you have all pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

1. [5] Draw a control-flow graph for the function in Figure 1. In addition to the usual CFG features, label each node with the corresponding code-line number.



2. [5] Statement Coverage: For each test in the Figure 2 test suite, list the nodes covered by the test.

Test 1) 2, 3, 4, 5, 8, 10

Test 2) 2, 3, 4, 5, 6, 8, 10

3. [2] Does the test suite achieve statement coverage? If not, what nodes did the test suite miss?

Yes

4. [5] Branch Coverage: For each test in the Figure 2 test suite, list the relevant edges (with respect to branch coverage) covered by the test case. Denote an edge like this, 2→3, which denotes the edge from node 2 to node 3.

Test 1) 4→5, 5→8, 4→10

Test 2) 4→5, 5→6, 4→10

5. [2] Does the test suite achieve branch coverage? If not, what relevant edges did the test suite miss?

Yes

6. [4] List all the paths through the CFG for the function in Figure 1. For each path, list the sequence of nodes on the path. You need only cover executions that involve at most 1 iteration of each loop (if there are any loops). There may be more lines below than there are paths; use only as many lines as are needed.

Path A) 2, 3, 4, 10

Path B) 2, 3, 4, 5, 8, 4, 10

Path C) 2, 3, 4, 5, 6, 8, 4, 10

Path D)

Path E)

7. [5] Path Coverage: For each test in the test suite, list the path covered by the test case.

Test 1) B

Test 2) C

8. [2] Does the test suite achieve path coverage? If not, what path(s) did the test suite miss?

No, path A is not covered

Bonus Problems

9. [5] Consider the buggy variant function in Figure 3. Which, if any, of the tests in Figure 2 would reveal this bug?

Test 1 would reveal the bug.
(Test 2 would not reveal it.)

10. [2] In general, which of statement, branch, and path coverage tend to require more tests to be written?

    a) statement > branch > path

    b) statement > path > branch

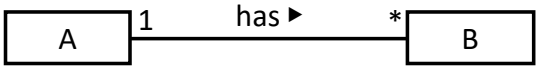
    c) branch > path > statement

    d) branch > statement > path

    e) path > statement > branch

    (f) path > branch > statement

11. [2] Circle the <u>two</u> answers that correctly express the following association relationship.



    a) Each A has one B

    (b) Each A has many Bs

    c) Each A belongs to one B

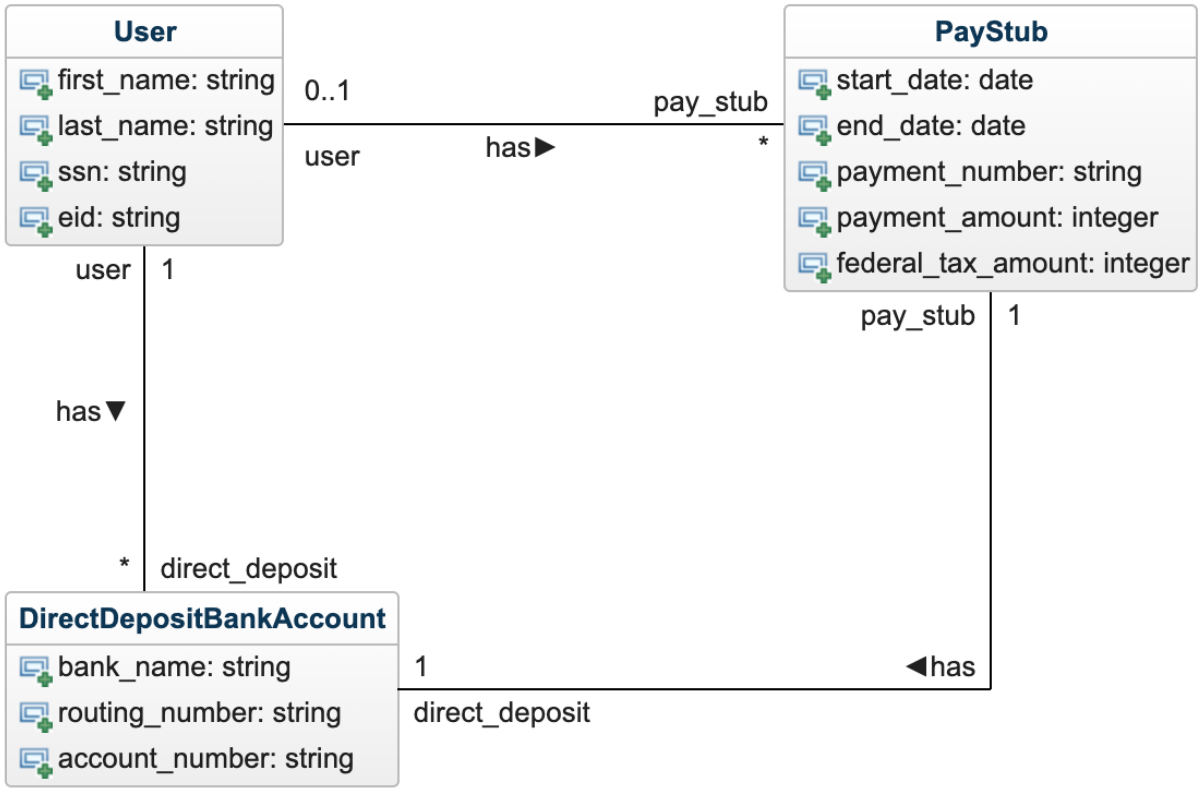
    d) Each B has one A

    e) Each B has many As

    (f) Each B belongs to one A

12. Imagine that you have been hired to build a payroll system. Create an object-oriented data model based on the following natural-language requirements. When deciding what to include, remember that the point here is that you are creating a design for your Rails MVC model. Your answer should take the form of a UML class diagram. Include only things that are specifically described.
    a) [6] Include all relevant classes and attributes.
    b) [6] Include all relevant associations and generalization relationships. Label all associations and association ends and include all multiplicities.

A user has a first name, last name, SSN (social security number), and EID (employee identification number). Each user has a set of pay stubs. Each pay stub has pay-period start and end dates, a payment number, a payment amount (in cents), and a federal tax amount (in cents). Each user also has a set of direct-deposit bank accounts. Each direct-deposit bank account has the name of the bank, the bank's routing number, and the user's bank-account number. Each pay stub is deposited in one of the user's direct-deposit bank accounts.



Figures

```
 1  def find_smallest(array)
 2      smallest = array[0]
 3      i = 1
 4      while i < array.length
 5          if array[i] < smallest
 6              smallest = array[i]
 7          end
 8          i = i + 1
 9      end
10      return smallest
11  end
```

Figure 1. A function that finds the smallest value in an array. The function has a precondition that the `array` argument must have a length of at least one.

Test #	Input	Expected Output
	`array`	
1	`[ 1, 2 ]`	`1`
2	`[ 2, 1 ]`	`1`

Figure 2. A test suite for the function in Figure 1.

```
 1  def find_smallest(array)
 2      smallest = array[1]
 3      i = 2
 4      while i < array.length
 5          if array[i] < smallest
 6              smallest = array[i]
 7          end
 8          i = i + 1
 9      end
10      return smallest
11  end
```

Figure 3. Buggy variant of the function from Figure 1 in which the author forgot that array indexes start at 0. Note that the bolded lines (2 and 3) contain the buggy code.