CSE 321 – Homework Assignment #4

Preamble

You will complete this homework in small groups, assigned randomly by Canvas. Please check Canvas for your group.

Turn in a PDF and other files to canvas by the deadline. It can be a scan of handwritten work, if that is your preference. However, it must be legible in that case!

You may post questions to the related Discussion board, but do not share any of your work/solutions.

Please review the course syllabus and first lecture for a reminder of what is acceptable collaboration (outside of your group) and what is not. Ask me if you are unclear.

Start early!

Contribution Declaration

You must describe each person's contribution as a separate section of the homework.

Office Hour Reminder

- Julius Higiro, higirodj@miamioh.edu , Monday 730pm-9pm Benton 001
- Liam Mosley, mosleylm@miamioh.edu , Tuesday 730pm-9pm Benton 002
- Dr. Stephan, matthew.stephan@miamioh.edu, Tuesdays and Thursdays 230-4

Questions

Question 1 (30 points)

Answer questions a–g for the graph defined by the following sets:

- $N = \{1, 2, 3, 4, 5, 6, 7\}$
- $N_0 = \{1\}$
- $N_f = \{7\}$
- $E = \{(1, 2), (1, 7), (2, 3), (2, 4), (3, 2), (4, 5), (4, 6), (5, 6), (6, 1)\}$

Also consider the following (candidate) test paths:

- $p_1 = [1, 2, 4, 5, 6, 1, 7]$
- $p_2 = [1, 2, 3, 2, 4, 6, 1, 7]$
- $p_3 = [1, 2, 3, 2, 4, 5, 6, 1, 7]$

a) Draw the graph. (4 points)

b) List the test requirements for Edge-Pair Coverage. (Hint: You should get 12 requirements of length 2 (3 nodes)) (6 points)

c) Does the given set of test paths satisfy Edge-Pair Coverage? If not, state what is missing. (2 points)

d) Consider the simple path **[3, 2, 4, 5, 6]** and test path **[1, 2, 3, 2, 4, 6, 1, 2, 4, 5, 6, 1, 7]**. Does the test path tour the simple path directly? With a sidetrip? If so, write down the sidetrip. (3 points)

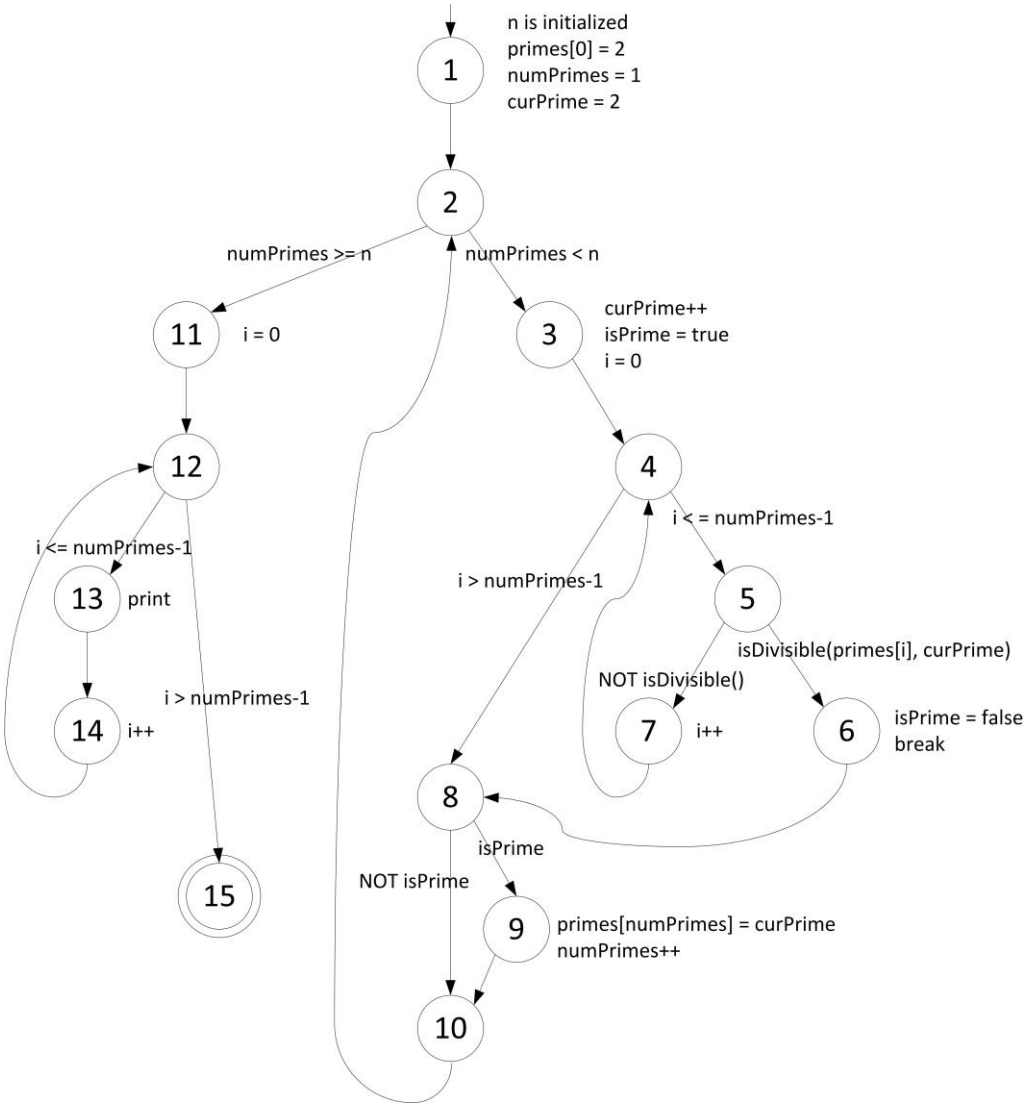
e) List the test requirements for Node Coverage (1 point), Edge Coverage (4 points), and Prime Path Coverage on the graph. (6 points)

f) List test paths from the given set that achieve Node Coverage but not Edge Coverage on the graph. (2 points)

g) List test paths from the given set that achieve Edge Coverage but not Prime Path Coverage on the graph. (2 points)

Question 2 (17 points)

Consider the method printPrimes() for questions a–e below from the compliable version on Canvas in the file PrintPrimes.java. A line-numbered version suitable for this exercise is available on Canvas as PrintPrimes.num. The following is a control flow graph for it. Use this graph to answer questions a-e.



(a) Consider test cases t_1 = (n = 3) and t_2 = (n = 5). Although these tour the same prime paths in printPrimes(), they do not necessarily find the same faults. Design a simple fault that t_2 would be more likely to discover than t_1 would. (2 points)

(b) For printPrimes(), find a test case such that the corresponding test path visits the edge that connects the beginning of the **while** statement to the *for* statement without going through the body of the while loop. (2 points)

(c) List the test requirements for Node Coverage (1 point), and Edge Coverage (9 points)

(d) List test paths that achieve Node Coverage but not Edge Coverage on the graph. (2 points)

(e) List test paths that achieve Edge Coverage but not Prime Path Coverage on the graph. (2 points)

Question 3 (12 points)

Use the following methods trash() and takeOut() to answer questions a–c. You can find the code on Canvas.

```
 1 public void trash (int x)
 2 {
 3    int m, n;
 4
 5    m = 0;
 6    if (x > 0)
 7       m = 4;
 8    if (x > 5)
 9       n = 3*m;
10    else
11       n = 4*m;
12    int o = takeOut (m, n);
13    System.out.println ("o is: " + o);
14 }
```

```
15 public int takeOut (int a, int b)
16 {
17    int d, e;
18
19    d = 42*a;
20    if (a > 0)
21       e = 2*b+d;
22    else
23       e = b+d;
24    return (e);
25 }
```

a) Give all call sites using the line numbers given. (2 points)

b) Give all pairs of *last-def*s and *first-use*s. (4 points)

c) Provide test inputs that satisfy *all-coupling-uses* (note that trash() only has one input). (6 points)

Question 4 (16 points)

Use the class BoundedQueue2 for questions a–f below. A compilable version is available on Canvas in the file BoundedQueue2.java. The queue is managed in the usual circular fashion.

Suppose we build a FSM where states are defined by the representation variables of **BoundedQueue2**. That is, a state is a 4-tuple defined by the values for [*elements, size, front, back*]. For example, the initial state has the value [[*null, null*], 0, 0, 0], and the state that results from pushing an object *obj* onto the queue in its initial state is [[*obj, null*], 1, 0, 1].

a) We do not actually care which specific objects are in the queue. Consequently, there are really just four useful values for the variable *elements*. What are they? (2 points)

b) How many states are there? (2 points)

c) How many of these states are reachable? (2 points) (you can answer this in part d)

d) Show the reachable states in a drawing. (4 points)

e) Add edges for the enQueue() and deQueue() methods. (For this assignment, ignore the exceptional returns, although you should observe that when exceptional returns are taken, none of the instance variables are modified.) (you can answer this in part d) (3 points)

f) Define a small test set that achieves Edge Coverage. Implement and execute this test set. You might find it helpful to write a method that shows the internal variables at each call. (3 points)