CITS5501 Software Testing and Quality Assurance
Semester 1, 2020

1. Consider the Javadoc documentation and signature for the following Java method, which searches inside an array of `char`s for a particular value.

(Adapted from the Android version of the Java standard library.)

```
/**
 * Performs a binary search for {@code value} in the ascending sorted array
 * in the range specified by fromIndex (inclusive) and toIndex (exclusive).
 * Searching in an unsorted array has an undefined result. It's also
 * is found if there are multiple occurrences of the same element.
 *
 * @param array the sorted array to search.
 * @param startIndex the inclusive start index.
 * @param endIndex the exclusive start index.
 * @param value the element to find.
 * @return the non-negative index of the element, or a negative index which
 *         is {@code -index - 1} where the element would be inserted.
 * @throws IllegalArgumentException if {@code startIndex > endIndex}
 * @throws ArrayIndexOutOfBoundsException if {@code startIndex < 0 ||
 * endIndex > array.length}
 * @since 1.6
 */
public static int binarySearch(char[] array, int startIndex, int endIndex,
    char value)
```

Discuss how you would go about creating tests using Input Space Partitioning. What steps are involved? What is the input domain? And what characteristics and partitions would you use?

List three different tests derived using this method.

Discuss how you would assess whether a set of tests have *base choice* coverage. What would you use for base choices?

2. Here is the body of the method.

```
1  public static int binarySearch(char[] array, int startIndex, int endIndex,
       ↪ char value) {
2      if (startIndex > endIndex) {
3          throw new IllegalArgumentException();
4      }
5      if (startIndex < 0 || endIndex > array.length) {
6          throw new ArrayIndexOutOfBoundsException();
7      }
8
9      int lo = startIndex;
10     int hi = endIndex - 1;
11     while (lo <= hi) {
12         int mid = (lo + hi) / 2;
13         char midVal = array[mid];
14         if (midVal < value) {
15             lo = mid + 1;
16         } else if (midVal > value) {
17             hi = mid - 1;
18         } else {
19             return mid;  // value found
20         }
21     }
22     return lo * -1;  // value not present
23 }
```

Discuss how you would construct a control-flow graph for the method. Try drawing the graph, stating any simplifying assumptions you need to make.

See if you can identify prime paths in which the loop is executed:

- zero times
- once
- more than once

See if you can construct test cases for these prime paths.

Exercises for self-study

3. Work out whether your tests from question two have any of the following sorts of coverage:

- node coverage
- edge coverage

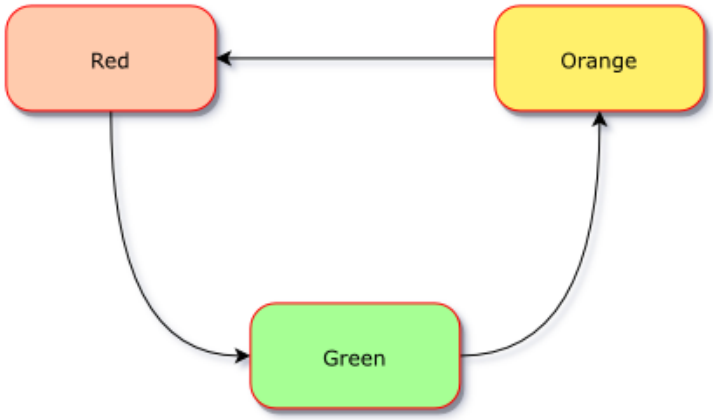
4. See if you can identify all the prime paths in the graph.

Does your set of tests have prime path coverage? Why, or why not?

5. *State diagrams* are used to give an abstract description of the behaviour of a stateful system. Take a look at the Wikipedia page on them: https://en.wikipedia.org/wiki/State_diagram. *States* are represented by nodes in a directed graph, and permissible *transitions* between states are represented by edges.

Example state diagrams:

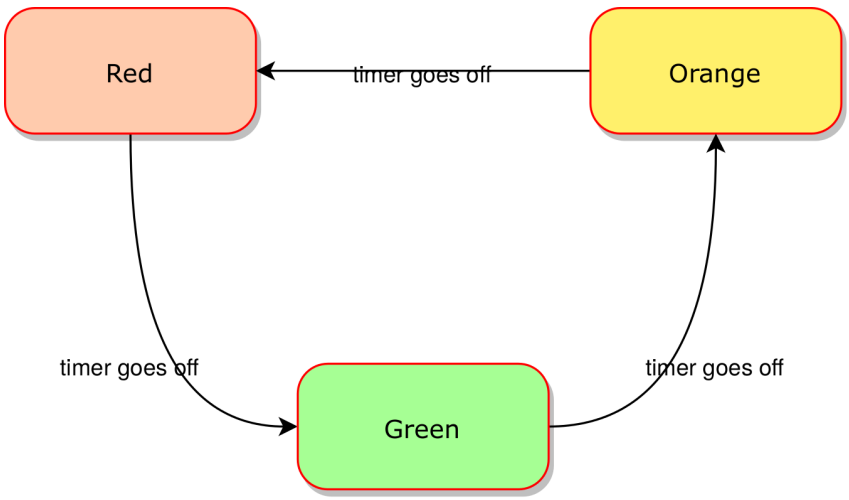
For example, the following state diagram could be used to represent a (very simple) traffic light system:



This diagram doesn't represent any information about when the traffic light changes state, or under what circumstances – just what the valid *state transitions* are.

In most systems, we want to represent what *events* will *trigger* transitions. (These events might arise from within the system – e.g., a timer going off – or from outside – e.g. a button being pressed by a user.) We can indicate events using labels on the node edges. (This is a simplified version of the syntax used for UML state charts.)

Suppose for our traffic light system that an internal timer goes off every 10 seconds, making the system change state. (This is a *very* fast-changing traffic light.) We could represent that as follows:



Exercises:

i. Draw a state diagram for a lift. The lift services a building with two floors. On each floor is a single button which call calls the lift to that floor. On the inside of the lift are two buttons, one which directs the lift to go to the ground floor, and another which directs the lift to go the first floor.

   What event types will be required? What states will be required?

ii. We can also add *guards* on each edge, which limit the circumstances in which a transition will occur. These are added in square brackets after the event name.

   Are any guards needed for our lift system?

3. If we define test cases for this system, what will be the inputs? And what will be the outputs?

   Can you define a set of test cases which give *edge coverage* of the lift system?