CS 4501/6501: In-class 12
Logic Coverage Criteria for Source Code

7-Nov-2017

Names:

Purpose: Understand and practice applying logic coverage to program source code
Instruction: Work with your neighbors in groups. Consider an implementation of the old engineering joke: Good, Fast, Cheap. Pick any two.

```
public final class GoodFastCheap {

    boolean good  = false;
    boolean fast  = false;
    boolean cheap = false;

    public void makeGood () {
       good = true;
       if (fast && cheap) { cheap = false; }
    }

    public void makeFast () {
       fast = true;
       if (good && cheap) { good = false; }
    }

    public void makeCheap () {
       cheap = true;
       if (fast && good) { fast = false; }
    }

    public void makeBad ()       { good = false; }
    public void makeSlow ()      { fast = false; }
    public void makeExpensive () { cheap = false; }

    public boolean satisfactory() {
       if ((good && fast) || (good && cheap) || (fast && cheap)) {
          return true;
       }
       return false;
    }

    public static void main(String[] args) {

    // Question:  How well do the following tests exercise the clauses?

       GoodFastCheap gfc = new GoodFastCheap(); // g f c
                          gfc.satisfactory(); // F F F
       gfc.makeGood();      gfc.satisfactory(); // T F F
       gfc.makeFast();      gfc.satisfactory(); // T T F
       gfc.makeCheap();     gfc.satisfactory(); // T F T
       gfc.makeSlow();      gfc.satisfactory(); // T F T
    }
}
```

- What does it mean to get RACC coverage on this code? How?

- Does the given sequence (the `main()` method) achieve RACC?

- What are the options for JUnit tests?

- What does it mean to get RACC coverage on a variation of `satisfactory()`:

```
public boolean satisfactory() {
   if (good && fast) return true;
   if (good && cheap) return true;
   if (fast && cheap) return true;

   return false;
}
```