CS 4501/6501: Exercise
Source Mutation

16-Nov-2017

Names:

Purpose: Understand and practice applying mutation testing to program source code
Instruction: Work with your neighbors in groups. Consider (again) an implementation of the old engineering joke: Good, Fast, Cheap.

```
public final class GoodFastCheap {

    // boolean variables good, fast, and cheap
    // and other stuff omitted

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

    public boolean isSatisfactoryRefactored() {
       if (good && fast)     return true;
       if (good && cheap)    return true;
       if (fast && cheap)    return true;
       return false;
    }
}
```

Consider a mutation operator that replaces boolaen variables with the constants "T" and "F" (Reminder: we are considering first-order mutation testing; i.e., only one change per mutant)

- How many mutants does this operator generate for
    - `isSatisfactory()`?
    - `isSatisfactoryRefactored()`?

- Some of these mutants are bound to be redundant. How many? Why?

- Consider a pair of corresponding mutants, one in each method. Will these mutants be killed by exactly the same tests? Is this a good thing?

- For each mutant, find all the tests that kill that mutant

- Analyze weak vs. strong mutation for these mutants. Is there any difference? Does your answer depend on any assumptions?

You might find the following table useful. You should add more columns as needed

```
p = ((g && f) || (g && c) || (f && c))
```

Row	g	f	c	Org												
1	T	T	T													
2	T	T														
3	T		T													
4	T															
5		T	T													
6		T														
7			T													
8																

```
p1 = (g && f)
p2 = (g && c)
p3 = (f && c)
```

Row	g	f	c	Org p1	Org p2	Org p3											
1	T	T	T														
2	T	T															
3	T		T														
4	T																
5		T	T														
6		T															
7			T														
8																	