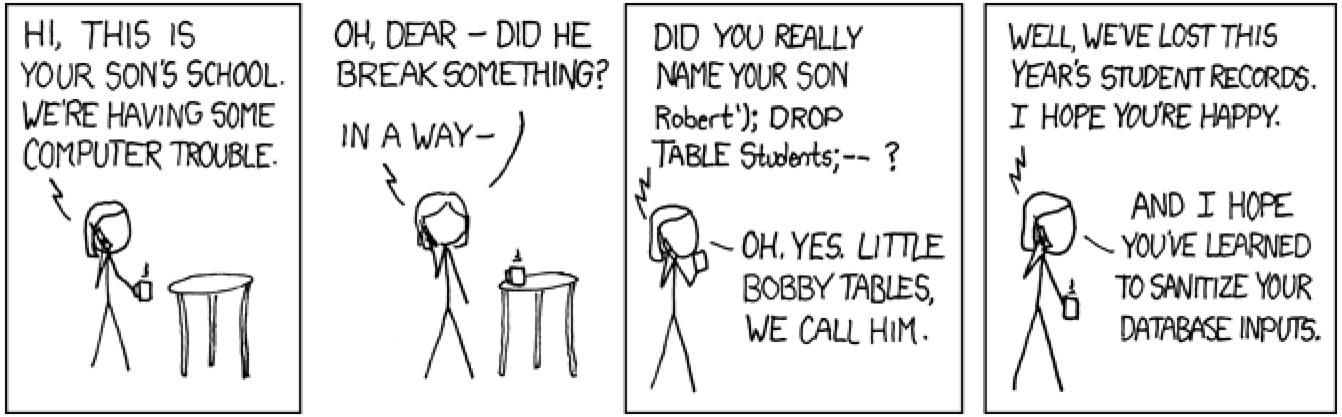
# CSc 110, Autumn 2016

## Lecture 22: Testing

Thanks to Atif Memon from UMD for disaster examples


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

# Assertion example 2

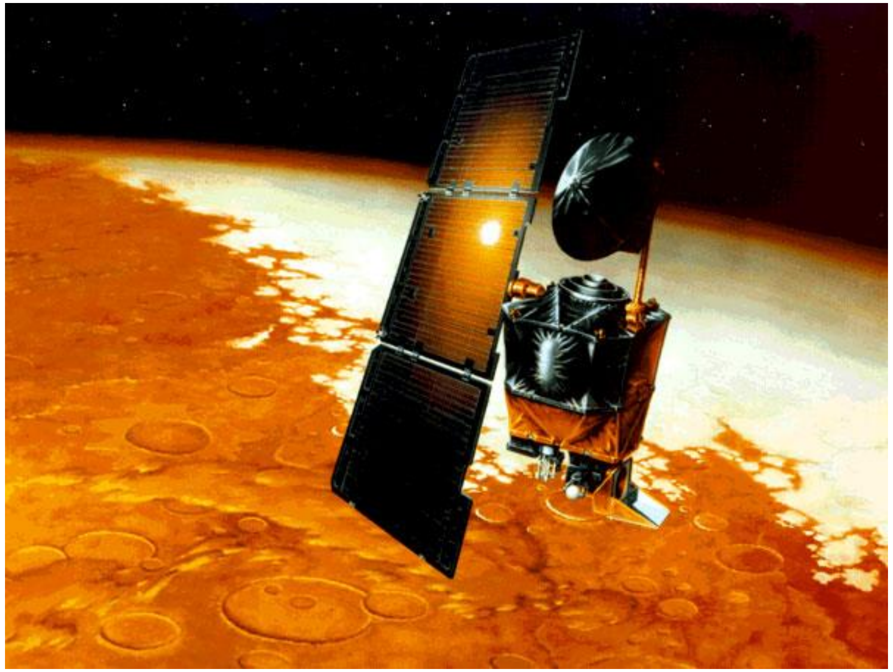
```
def mystery():
    prev = 0
    count = 0
    next = input()
    # Point A

    while (next != 0):
        # Point B

        if (next == prev):
            # Point C
            count += 1

        prev = next
        next = input()
        # Point D
    # Point E
    return count
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

| | next == 0 | prev == 0 | next == prev |
|---|---|---|---|
| Point A | SOMETIMES | ALWAYS | SOMETIMES |
| Point B | NEVER | SOMETIMES | SOMETIMES |
| Point C | NEVER | NEVER | ALWAYS |
| Point D | SOMETIMES | NEVER | SOMETIMES |
| Point E | ALWAYS | SOMETIMES | SOMETIMES |

# Why talk about testing?

- Mars Climate Orbiter
  - Purpose: to relay signals from the Mars Polar Lander once it reached the surface of the planet
  - Disaster: smashed into the planet instead of reaching a safe orbit
  - Why: Software bug - failure to convert English measures to metric values



# Why talk about testing?

- THERAC-25 Radiation Therapy
  - 1986: two cancer patients at the East Texas Cancer Center in Tyler received fatal radiation overdose
  - Why: Software bug - mishandled race condition (i.e., miscoordination between concurrent tasks)

# Why talk about testing?

- London Ambulance Service
  - Purpose: automate many of the human-intensive processes of manual dispatch systems associated with ambulance services in the UK – functions: Call taking
  - Failure of the London Ambulance Service on 26 and 27 November 1992
    - Load increased,  emergencies accumulated, system made incorrect allocations

# Industry comparison

- “If the automobile industry had developed like the software industry, we would all be driving $25 cars that get 1,000 miles to the gallon.”
- “Yeah, and if cars were like software, they would crash twice a day for no reason, and when you called for service, they’d tell you to reinstall the engine.”
- Now days you can get a job as a software engineer developing cars!

# Black box testing

- Testing the program does what the specification requires
- Tester has no access to the code
  - Sometimes doesn't even need to know how to code

# White box testing

- Examining code for potential problems
- Makes sure code meets specification
- Requires programming knowledge

# Test driven development

- A software engineering philosophy
- Tests are written before the code is written

# Types of testing

- Unit test: verifies correctness of a small piece of testable code in isolation
- Integration test: verifies different small already tested components of the program work together correctly
- Regression testing: a complete retesting of a modified program
- Stress testing: tests the behavior under peak user volumes
- Performance, security, usability and many more

# Black box testing example

Write a function named `remove_bad_pairs` that accepts as a parameter a list of integers, and removes any adjacent pair of integers in the list if the left element of the pair is larger than the right element of the pair. Every pair's left element is at an even-numbered index in the list, and every pair's right element is at an odd index in the list. For example, suppose a variable named `list` stores the following element values:

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

We can think of this list as a sequence of pairs:

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

The pairs 9-2, 8-5, 6-3, and 3-1 are "bad" because the left element is larger than the right one, so these pairs should be removed. So the call of `remove_bad_pairs(list)` would change the list to store:

```
[3, 7, 5, 5, 4, 7]
```

If the list has an odd length, the last element is not part of a pair and is also considered "bad;" it should therefore be removed by your function.

If an empty list is passed in, the list should still be empty at the end of the call.

# What are the invariants at the end of the function?

- The list will have an even length
- For each pair the left element will be less or equal to the right

# White box testing example

```
remove_bad_pairs(list):
    if (len(list) % 2 != 0):
        v.remove(list[-1])
    i = 0
    while(i < len(list)):
        if (i % 2 != 0 and list[i - 1] > list[i]):
            list.remove(i - 1)
            list.remove(i)
```

# What should you test?

- Focus on edge cases
    - Don't just test 1, 2, 3, 4, 5, 6, etc – that won't tell you anything new

- Common edge cases
    - 1, 0, -1
    - Empty list

# White box test a more complex problem

- Consider the following Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

- Write a program that displays any movies containing a phrase:

```
Search word? part

Rank    Votes   Rating  Title
2       139085  9.0     The Godfather: Part II (1974)
40      129172  8.5     The Departed (2006)
95      20401   8.2     The Apartment (1960)
192     30587   8.0     Spartacus (1960)
4 matches.
```