Decision Tables.
Monday/Wednesday Classes

- Jim is traveling Monday-Wednesday next week (Feb. 4-6)
  - Principal Investigator’s meeting for a grant

- Cameron Alston will be leading the class

- Monday:
  - Play a board game that is new to you
  - Take notes

- Wednesday
  - Play further
  - With a partner, in-class, write a short 1-2 page analysis of some aspect of gameplay

- **Attendance is required Monday and Wednesday**
Team Meetings

- Will be meeting
  - Siegebreakers, 3:30pm, today (Friday)
  - Lens, 4:30pm, today (Friday)

- No team meetings
  - Monday, Tuesday, Wednesday of next week

- No team meetings with:
  - Militarium, Biogenesis, Tearable World, 10 Days Left to Live, Pixture, Asterouge

- Will meet We’re Screwed! On Thursday, February 7
Upcoming deadlines

- **Today**
  - Sprint 1 report due
  - Sprint 2 plan due

- **Today, Friday (Feb. 1): team status reporting**
  - Due by midnight
  - Report on team activities this week
  - Be sure to use team status reporting template
    - courses.soe.ucsc.edu/courses/cmps171/Winter13/01/pages/teamstatus-template
Upcoming events

- Winter Job & Internship Fair
  - Tuesday, February 5
  - See [http://careers.ucsc.edu/](http://careers.ucsc.edu/) for more information
Lab Cleanup Schedule

- This week: Tearable World
- Next week: We’re Screwed!

- Team duties:
  - Ensure overflowing trash cans are emptied to bin outside in 3rd floor courtyard (anytime during week)
  - By 5pm Monday and 5pm Friday (unless things get out of control, then more often):
    - Pick up food containers, bottles, etc.
    - Pick up stray craft materials, pens, etc and return to drawers
    - Clean off tables in conference rooms and big circular table
    - Report any major soda/food spills to me, so we can call cleanup crews
    - Put controllers/game boxes/etc. away (tidy up game area)
    - Report any cleaning materials needed
Sprint Report

- The output of a Sprint Retrospection meeting
  - Meeting held after the end of every sprint
  - Provides time to reflect on what happened, determine improvements

- Questions to be answered:
  - What things should we stop doing?
    - Actions/activities the team did that were harmful, hurt progress
  - What should we start doing?
    - Actions/activities that will improve how we do work
  - What is working well that we should continue to do?
    - What is working well, and shouldn’t be touched?
  - What work was completed, and not completed?
    - User stories that were completed, not completed
  - What is our rate of completing work?
    - Story points completed, ideal work hours completed
    - Story points/day, ideal work hours/day
    - Average story points/day & ideal work hours/day across all sprints
    - Final sprint burndown chart
Triangle problem (revisited)

- **Triangle problem**
  - A program reads three integer values from the console. The three values are interpreted as representing the lengths of the sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles, or equilateral.

- **Recall:**
  - **Equilateral triangle:**
    - Three equal sides, three equal angles (all 60 degrees)
  - **Isosceles triangle:**
    - Two equal sides, two equal angles
  - **Scalene triangle:**
    - No equal sides, no equal angles

- **Use Output classes**
  - In this case, is best to develop test cases based on output equivalence classes
  - What are they?
  - What are examples of strong normal and weak robust test cases (assume max side length of 200)
Decision Table Testing

- Core idea:
  - Use a table format to capture the relationship between input conditions and output actions
  - Then, use combination of conditions and actions to develop test cases
  - Benefit: tables provide a rigorous way of specifying conditions and actions, since omissions are very clear (an empty part of the table)

- Decision table format:
Decision Table Format

- **Conditions** describe logical states the software can be in
  - Often describe ranges of inputs
- **Actions** describe possible things the software can do
  - Often are the outputs
- **Rules** describe what actions (if any) occur for a given set of conditions

<table>
<thead>
<tr>
<th>Conditions (condition stub)</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>…</th>
<th>Rule n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions (action stub)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Triangle Problem as Decision Table

<table>
<thead>
<tr>
<th>C1: a, b, c form a triangle?</th>
<th>F</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2: a=b?</td>
<td>--</td>
<td>--</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>C3: a=c?</td>
<td>--</td>
<td>--</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>C4: b=c?</td>
<td>--</td>
<td>--</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>A1: Not a triangle</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2: Scalene</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3: Isosceles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4: Equilateral</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5: Impossible</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Fill-in last 5 columns
## Triangle Problem as Decision Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generating Test Cases from Decision Table

- For each row in the table, pick actual values that satisfy the conditions, and then verify the output state(s) match the actions.

- In-class:
  - Find example test cases from the complete decision table for the triangle problem
  - What are the pros and cons of these test cases?
Random Testing

- One final black box testing technique is to use a random number generator to create inputs.
- Requires a stopping criteria
  - Could generate random numbers forever – how many do you want?
- Pro:
  - Can make a lot of tests, cheaply
  - Explore larger, variable part of input space
- Con:
  - Need to control seed to ensure test cases are repeatable (otherwise vary with each run)
  - Tests are more computationally expensive to run
  - Tests may require more work to create for the first time

Example: random testing the triangle problem
- Randomly pick triangle side values
- Stopping criteria: continue until every possible output has been seen at least once
- Example (from book):
  - 1289 test cases, of which 663 are nontriangles, 593 are scalene, 32 are isosceles, and 1 is equilateral
Oracles

- No, not the database company
- Think Oracle of Delphi
  - Tells the truth about your future

- A **test oracle** is a function that determines whether the program’s output for a given set of inputs is correct or incorrect.

- A human being often acts as the test oracle.
  - Either manually testing code, or determining what the outputs should be for an automatically run regression test with static inputs
- For random testing, the test oracle must be a computational function.
  - It can often be as difficult to make the test oracle as it is to create the test