Problem Solving and Programming

- Design
  - Requires intense concentration
  - When is the best time to fix bugs?
- Testing
  - Requires a lot of skill, practice
  - How does problem solving relate to testing?

Debugging Example #1

A man who has had a heart attack goes every evening to a supervised exercise program. He handles the exercise well during the first 15 sessions, maintaining a heart rate at about 100 beats/minute. In the middle of the 16th session, however, his heart rate suddenly shoots up to 130 beats/minutes. Although this may not be dangerous, nevertheless, the attendant has him stop exercising and calls the supervising doctor. The man is short of breath but otherwise feels fine. The change in heart rate appears to be his only symptom. What question(s) should the doctor ask?

Debugging Example #2

A man went to wash his face on awakening and found that there was no hot water. He knew to look for a special feature. He asked his wife whether she had done anything the day before near the boiler. Her response was in the negative. She added, however, “I didn’t have a chance to tell you, but the oil company sent a many yesterday to clean the furnace.” That certainly looked like a promising hint. A call to the oil company led to the solution of the problem.

Debugging

- One of the hardest parts of programming
- Strategy 1: Avoid bugs in the first place
  - Careful design (clean decomposition)
  - Care with syntactic issues (layout, commenting)
- Strategy 2: Implement in a series of small steps, and test along the way
  - This localizes new bugs to what changed in the program to introduce the bug.
- Finding bugs requires a disciplined, deductive approach

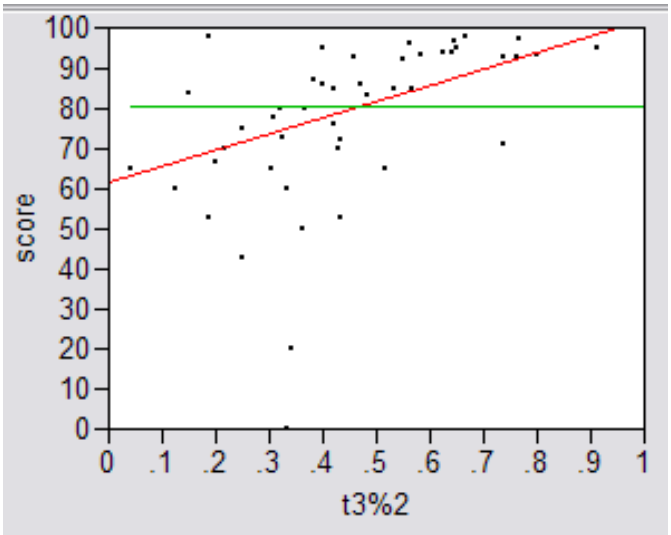
Scheduling

- Managing large-scale projects involves significant efforts to plan and schedule activities
  - It is human nature to work better toward intermediate milestones.
- The same concepts can/should be applied to mid-sized projects encountered in class.
  - For any project that needs more than a week of active work to complete, break into parts and design a schedule with milestones and deliverables.

Real Results #1

- CS2606, Fall 2006
- 3-4 week projects
- Kept schedule information:
  - Estimated time required
  - Milestones, estimated times for each
  - Weekly estimates of time spent.

Real Results #2



Real Results #3

- Results were significant:
  - 90% of scores below median involved students who did less than 50% of the project prior to the last week.
  - Few did poorly who put in > 50% time early
  - Some did well who didn't put in >50% time early, but most who did well put in the early time
- Correlations:
  - Strong correlation between early time and high score
  - No correlation between time spent and score
  - No correlation between % early time and total time

What is the Mechanism?

- Correlations are not causal
  - Do they behave that way because they are good, or does behaving that way make them good?
- Spreading projects over time allow the “sleep on it” heuristic to operate
- Avoiding the “zombie” effect makes people more productive (and cuts time requirements)

Myers-Briggs and Programming

- How do you think the personality dimensions relate to programming?
  - Extrovert: Act/reflect/act. Energy from activity.
    Introvert: Reflect/act/reflect. Activity requires downtime
  - Sensing: Method, informed from outside, build pattern from facts
    Intuition: Insight, informed from inside, fit facts to pattern
  - Thinking: Decision from logic, impersonal
    Feeling: Decision from harmony, personal
  - Judging: Planned, decided, fixed, on time
    Perceiving: Improvised, open, adaptable, dislike deadlines

Literature Results 1

- Huge differences in performance for programming time, debugging time, efficiency of resulting code. Why?
- Each task (design, implementation, testing, debugging) requires different skills
- Several studies done on relationships between MBTI and various aspects of programming

Literature Results 2

- We know that the distribution for MBTI among software engineers is different from the general population. (Does it matter?)

ISTJ	ISFJ	INFJ	INTJ
11.6%	13.8%	1.5%	2.1%
ISTP	ISFP	INFP	INTP
5.4%	8.8%	4.4%	3.3%
ESTP	ESFP	ENFP	ENTP
4.3%	8.5%	8.1%	3.2%
ESTJ	ESFJ	ENFJ	ENTJ
8.7%	12.3%	2.5%	1.8%

ISTJ	ISFJ	INFJ	INTJ
24%	2%	1%	7%
$R = 2.06$	$R = 0.14$	$R = 0.68$	$R = 3.40$
ISTP	ISFP	INFP	INTP
8%	5%	2%	8%
$R = 1.49$	$R = 0.57$	$R = 0.46$	$R = 2.46$
ESTP	ESFP	ENFP	ENTP
8%	1%	3%	7%
$R = 1.87$	$R = 0.12$	$R = 0.37$	$R = 2.19$
ESTJ	ESFJ	ENFJ	ENTJ
15%	4%	1%	4%
$R = 1.73$	$R = 0.33$	$R = 0.41$	$R = 2.23$

Literature Results 3

- Code-review task (bug fixing)

	F	T
N	8.71	9.10
S	4.27	6.62

5 Habits of Highly INeffective Programmers

1. Design with less than total focus
2. Disorganized code
   - Style, comments, design
3. Bite off more than you can chew
   - During impementation
4. Debug in a random walk
5. Program/debug In zombie mode
   - a.k.a Don’t start early enough