Productivity

"Meanwhile, obsessing about productivity is way up."

The Story So Far …

- We want to deliver and support a quality software product
- Software processes are carried out by humans
  - Humans have biases
- Some humans are more productive than others at software engineering activities
  - How can we understand and improve such human expertise?

One-Slide Summary

- Humans demonstrate different levels of **expertise** (i.e., different productivity rates) at programming tasks.
- We consider a number of hypotheses, including hardware support, slow programmers and programs, abstractions, decompositions, and neural activity. For each, we examine relevant scientific literature.
- Organizations can provide hardware support. Individuals can practice abstractions and decompositions.

Outline, Psychology

- Real-Time Exercise
- Reading Discussion
  - Rapid Response Time
  - Programming Performance
  - Mythical Man-Month
  - Expertise in Problem Solving
  - Expert Bodies, Expert Minds
  - What Predicts Software Developers' Productivity?
- Advice

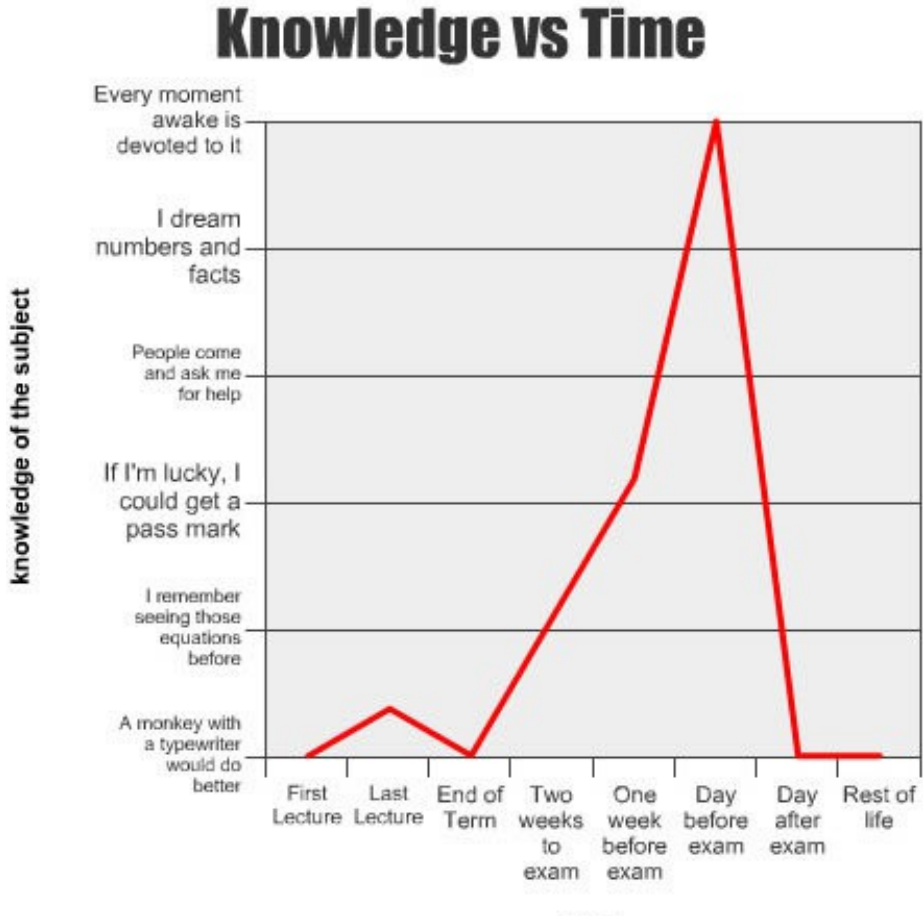
Real-Time Exercise

https://dijkstra.eecs.umich.edu/eecs483/shibboleth/productivity/

- You will be asked to solve a simple problem.
  - ***Get the correct answer as quickly as possible.***
  - ***This counts as the Participation if you submit an answer and explanation by midnight.***
- You will be timed (once you click "start").
- You can use *any* program, language or tool available to you.
- Once you have submitted your answer, you must briefly explain what you did.
- I will cut things off after ~10 minutes.

Distribution of Times

- How many different tasks were students given?
- What did you observe, roughly, as the range and variance of times?

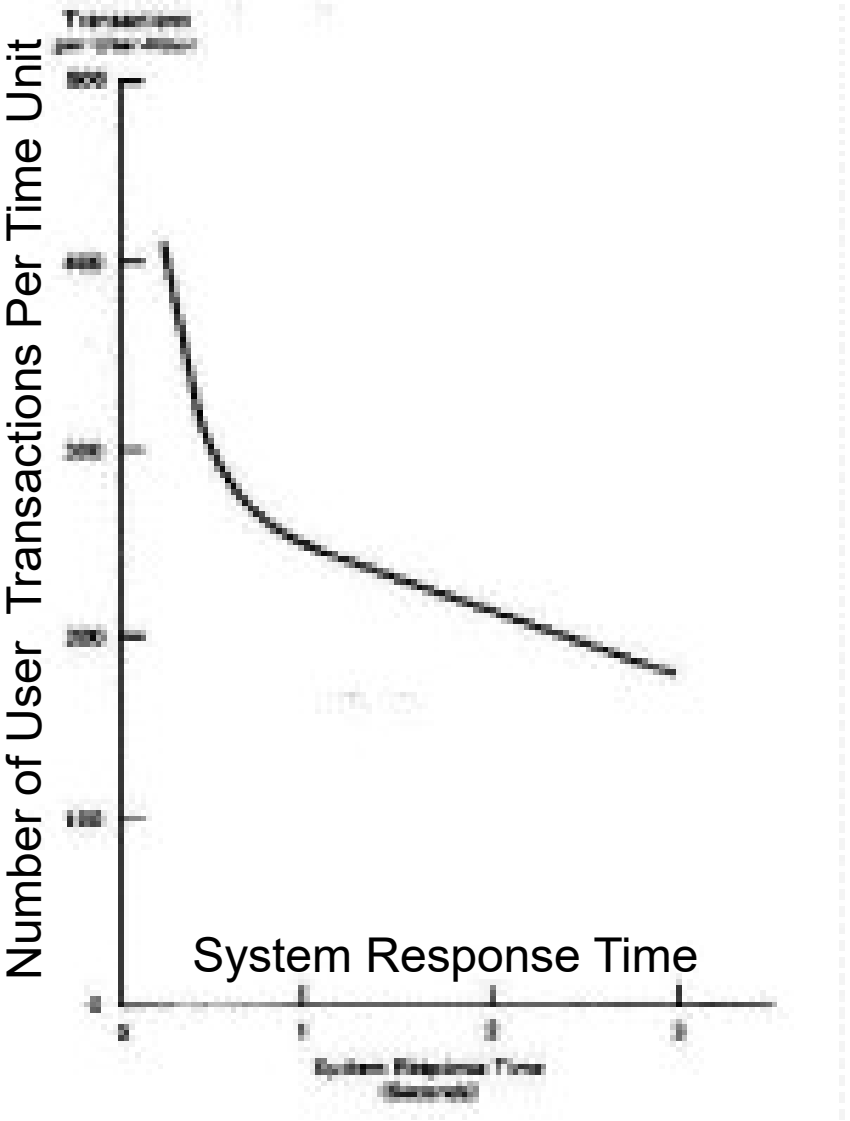


Hypotheses

- My computer is slow.
- I'm slow and so is my program.
- I picked the wrong language/abstraction and couldn't break up the problem.
- I did not recognize the true components of the problem.
- My brain is currently inefficient, requiring much metabolism for little neural activation.

Rapid Response Time

- Walter Dougherty and Ahrvind Thadani. *The Economic Value of Rapid Response Time*. IBM Systems Journal, 1982.
  - Read chart "backward", from Right to Left.
  - Productivity goes up, then sharply up.

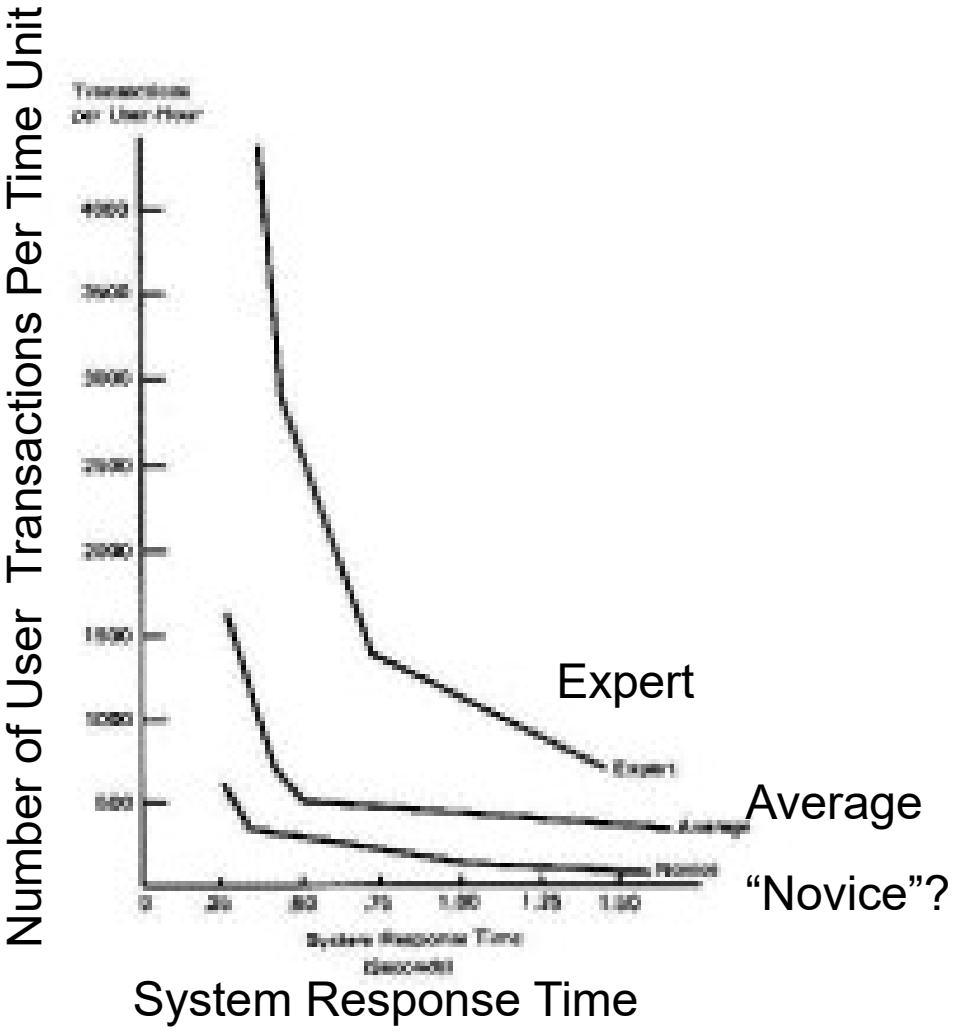


Rapid Response Time

"...each second of system response degradation leads to a similar degradation added to the user's time for the following [command]. This phenomenon seems to be related to an individual's attention span. The traditional model of a person thinking after each system response appears to be inaccurate. Instead, people seem to have a sequence of actions in mind, contained in a short-term mental memory buffer. Increases in SRT [system response time] seem to disrupt the thought processes, and this may result in having to rethink the sequence of actions to be continued."

Rapid Response Time

- Figure 7



Rapid Response Time

- The SPD study measured 75 work sessions of 15 engineers at graphic display terminals as they performed various physical design tasks. Their transaction rate data confirmed Thadhani's curve, (Figure 7). Indeed, it showed considerably more. All users benefited from sub-second response time. In addition, on average, experienced engineer working with sub-second response was as productive as an expert with slower response. A novice's performance became as good as the experienced professional and the productivity of the expert was dramatically enhanced.

Rapid Response Time

- Example implication, from the reading:

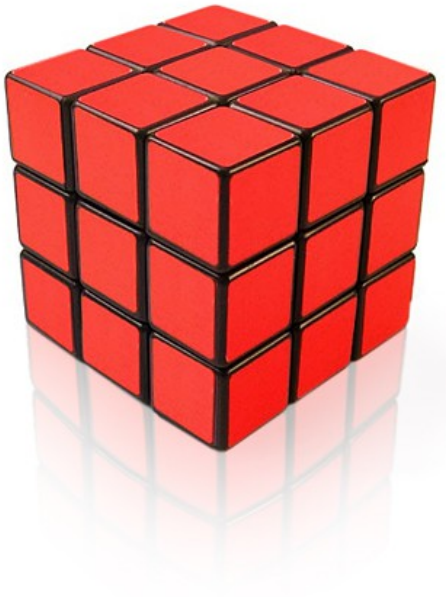
  "The system and user cost for this time were estimated at $900,000 monthly (Figure 6), 15 times the incremental cost of a new processor capable of providing sub-second response time to 500 simultaneous users. For the National Institutes of Health, the cost of upgrading their processor was more than justified by the savings in user time and the restoration of their low task costs.

  The engineers use display terminals specifically designed for the high transaction rates necessary to manipulate graphic images."

Programming Performance

- H. Sackman, W. J. Erikson and E. E. Grant. *Exploratory Experimental Studies Comparing Online and Offline Programming Performance*. Communication of the ACM, 1968.

- Summary?



Programming Performance

TABLE III. RANGE OF INDIVIDUAL DIFFERENCES IN PROGRAMMING PERFORMANCE

Performance measure	*Poorest score*	*Best score*	*Ratio*
1. Debug hours Algebra	170	6	28:1
2. Debug hours Maze	26	1	26:1
3. CPU time Algebra (sec)	3075	370	8:1
4. CPU time Maze (sec)	541	50	11:1
5. Code hours Algebra	111	7	16:1
6. Code hours Maze	50	2	25:1
7. Program size Algebra	6137	1050	6:1
8. Program size Maze	3287	651	5:1
9. Run time Algebra (sec)	7.9	1.6	5:1
10. Run time Maze (sec)	8.0	.6	13:1

Programming Performance

TABLE I. EXPERIENCED PROGRAMMER PERFORMANCE

DEBUG MAN-HOURS

	Algebra		*Maze*	
	Online	*Offline*	*Online*	*Offline*
Mean	34.5	50.2	4.0	12.3
SD	30.5	58.9	4.3	8.7

CPU TIME (sec)

	Algebra		*Maze*	
	Online	*Offline*	*Online*	*Offline*
Mean	1266	907	229	191
SD	473	1067	175	136

Programming Performance

- A substantial performance factor designated as “programming speed,” associated with faster coding and debugging, less CPU time, and the use of a higher order language.
  - WRW: This is new, but not the whole story.
- A well-defined “program economy” factor marked by shorter and faster running programs, associated to some extent with greater programming experience and with the use of machine language rather than higher order language.
  - WRW: Similar explanation to the previous paper.

Programming Performance

- “Data were gathered on the subject's grades in the SDC programmer training class … and they were also given the Basic Programmer Knowledge Test. Correlations between all experimental measures, adjusted scores, grades, and the BPKT results were determined. … The results showed no consistent correlation between performance measures and the various grades and test scores.”

Programming Performance

- “It is apparent from the spread of the data that very substantial savings can be effected by successfully detecting low performers. Techniques measuring individual programming skills should be vigorously pursued ...”

- Why do CS companies use Skill-Based Interviews instead of just using your class grades?
  - See other lecture!

Fault Localization Accuracy

- Zachary P. Fry, Westley Weimer: *A Human Study of Fault Localization Accuracy.* International Conference on Software Maintenance (ICSM) 2010

TABLE II
PARTICIPANT SUBSETS AND AVERAGE ACCURACIES. THE COMPLETE HUMAN STUDY INVOLVED $n = 65$ PARTICIPANTS.

Subset	Average Accuracy	Number of Participants
All	46.3%	65
Accuracy $> 40\%$	55.2%	46
Experience > 4 years	51.5%	34
Experience ≥ 4 years	49.9%	51
Experience $= 4$ years	46.7%	17
Experience < 4 years	33.4%	14
Baseline: Guess Longest Line	6.3%	-
Baseline: Guess Randomly	<5.0%	-

The Mythical Man-Month

- Frederick Brooks. *The Mythical Man-Month*. Addison-Wesley, 1975/1995.

- Summary?

> Since software construction is inherently a systems effort—an exercise in complex interrelationships—communication effort is great, and it quickly dominates the decrease in individual task time brought about by partitioning. Adding more men then lengthens, not shortens, the schedule.

The Mythical Man-Month

- Brooks: SE is non-partitionable.

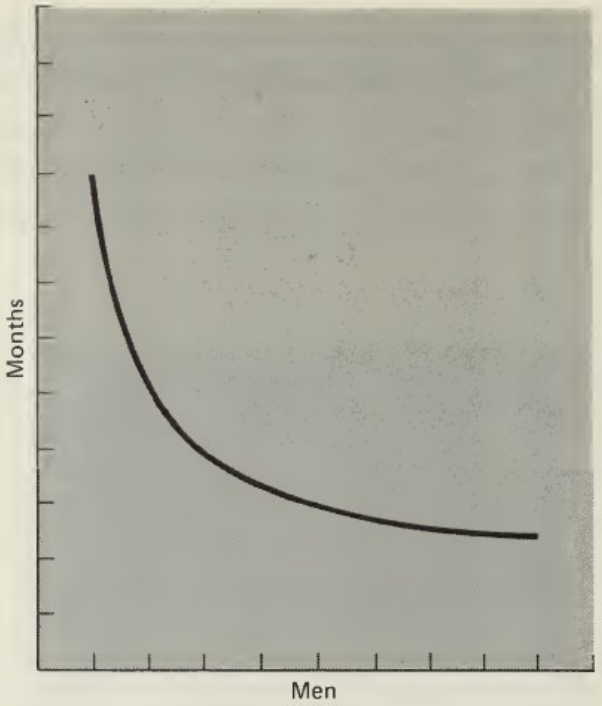


Fig. 2.3 Time versus number of workers—partitionable task requiring communication

Fig. 2.4 Time versus number of workers—task with complex interrelationships

The Mythical Man-Month

For some years I have been successfully using the following rule of thumb for scheduling a software task:

⅓ planning
⅙ coding
¼ component test and early system test
¼ system test, all components in hand.

This differs from conventional scheduling in several important ways:

1. The fraction devoted to planning is larger than normal. Even so, it is barely enough to produce a detailed and solid specification, and not enough to include research or exploration of totally new techniques.
2. The *half* of the schedule devoted to debugging of completed code is much larger than normal.
3. The part that is easy to estimate, i.e., coding, is given only one-sixth of the schedule.

The Mythical Man-Month

Corbató's Data

Both Harr's data and OS/360 data are for assembly language programming. Little data seem to have been published on system programming productivity using higher-level languages. Corbató of MIT's Project MAC reports, however, a mean productivity of 1200 lines of debugged PL/I statements per man-year on the MULTICS system (between 1 and 2 million words).[10]

But Corbató's number is *lines* per man-year, not *words*! Each statement in his system corresponds to about three to five words of handwritten code! This suggests two important conclusions.

- Productivity seems constant in terms of elementary statements, a conclusion that is reasonable in terms of the thought a statement requires and the errors it may include.[11]
- Programming productivity may be increased as much as five times when a suitable high-level language is used.[12]

The Mythical Man-Month

- 1200 lines / year = 3 lines of code per day
  - ***What?***
- Recall: “debugged code”
  - This includes coding, testing, debugging, etc.
  - Basically the entire software lifecycle
- More modern estimates: 10 LOC / day
- The real insight is the observation of **language invariance**.
  - You can get 10 lines of ASM or 10 lines of Python.

Trivia: Names

- Originally called Catholepistemiad, *this* institution was established in 1817. Its board of regents was formed later in 1837. However, a local justice called that name “neither Greek, Latin, nor English, [but merely] a piece of language gone mad.” At a speech there in 1960, President Kennedy announced his intention to establish the Peace Corps.

Trivia: Poetry

- Name the reclusive American poet and Amherst graduate associated with these works:
  - Because I could not stop for Death

    He kindly stopped for me
  - I'm nobody! Who are you?

    Are you nobody, too?
  - Tell all the Truth but tell it slant —

    Success in Circuit lies
  - My Life had stood — a Loaded Gun —

    In Corners — till a Day

Trivia: Gaming Metrics

- This term refers to the rate at which video game players can select units or otherwise issue orders. It is primarily associated with real-time strategy and fighting games such as StarCraft; a high value for this metric is associated with skill and expertise:
  - Beginner: ~50
  - Professional: ~300
  - Competition: ~400+

Trivia: Cuisine

- This fresh cheese is common in South Asia, especially in India. It is a non-melting, acid-set farmer cheese made by curdling heated milk with lemon juice or vinegar or yogurt, separating out the excess water, and cooling. It is commonly used in dishes in India, Nepal, Bangladesh and Pakistan.

Expertise in Problem Solving

- M. Chi, R. Glaser and E. Rees. *Expertise in Problem Solving*. Advances in the Psychology of Human Intelligence, 1982.

- Summary?

Expertise in Problem Solving

- "Both expert and novice proceed to solution by evoking the appropriate physics equations and then solving them. The expert often does this in one step, however …"
- "The speed with which a problem can be solved depends a great deal on the skill of the individual. Simon and Simon noted a 4:1 difference … Larkin also reported a similar difference between her experts and novices."

Expertise in Problem Solving

- "Another interesting aspect of novice problem solving is not only that they commit more errors than experts but that, even when they do solve a physics problem correctly, their approach is quite different."

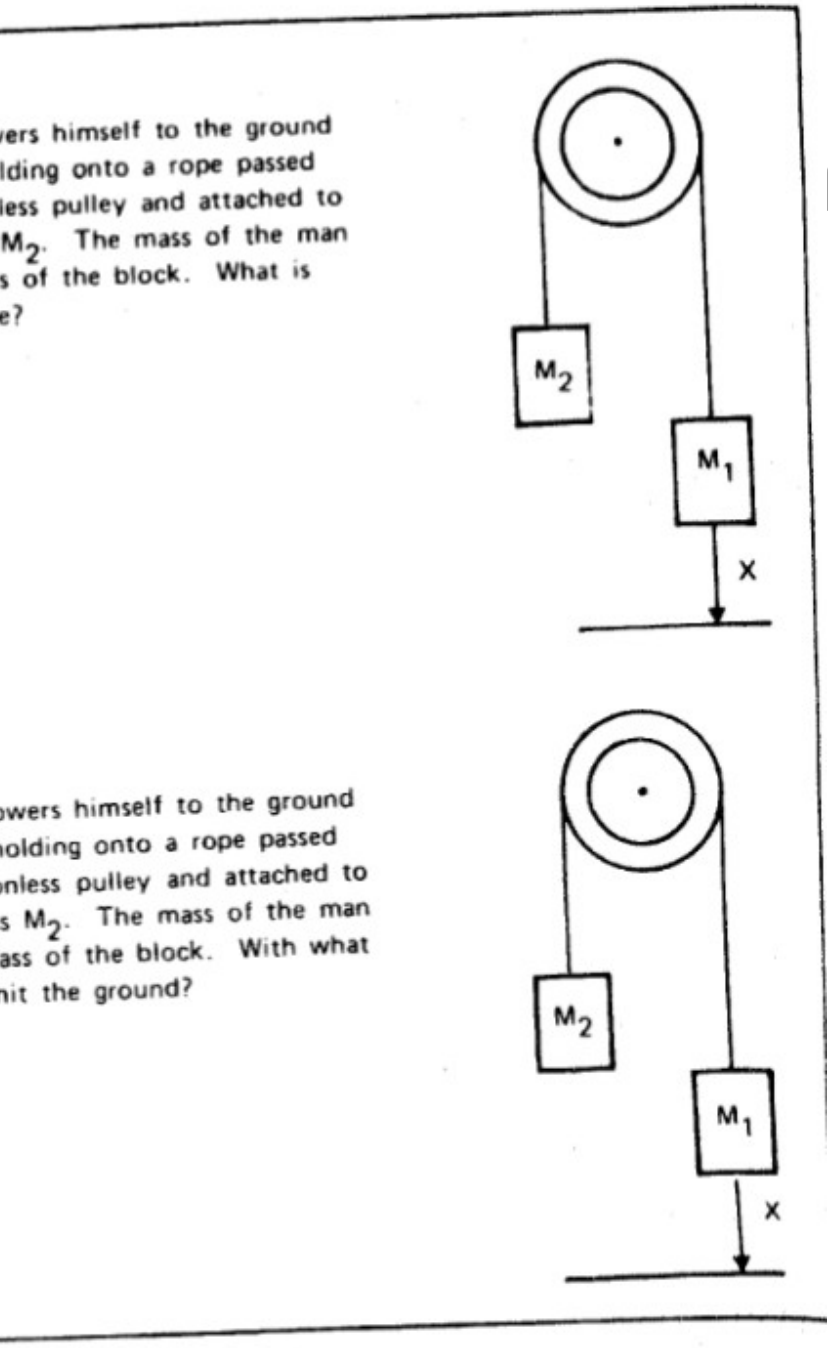


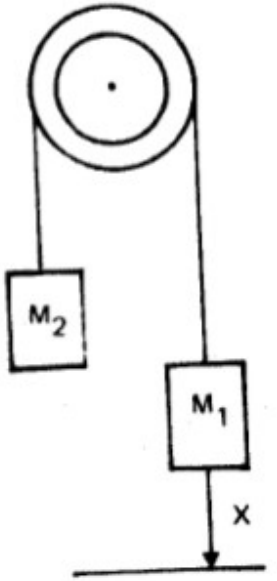
FIG. 1.6. Sample problems.

Problem Solving

- These two problems have a similar superficial structure

No. 11 (Force Problem)

A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. What is the tension on the rope?



No. 18 (Energy Problem)

A man of mass M_1 lowers himself to the ground from a height X by holding onto a rope passed over a massless frictionless pulley and attached to another block of mass M_2. The mass of the man is greater than the mass of the block. With what speed does the man hit the ground?

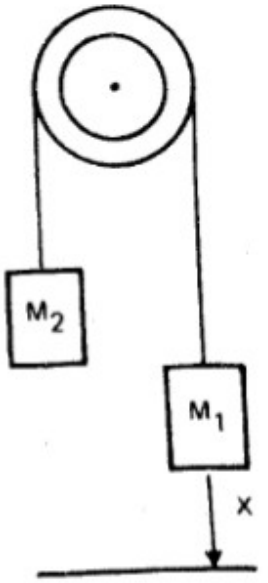
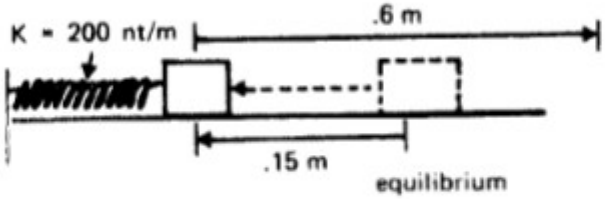


FIG. 1.6. Sample problems.

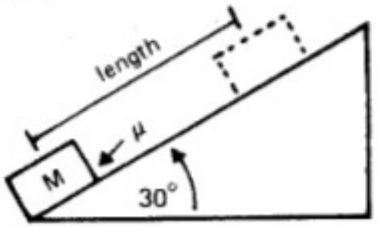
Expertise in Problem Solving

Diagrams Depicted from Problems Catergorized by Experts within the Same Groups

Problem 6 (21)



Problem 7 (35)



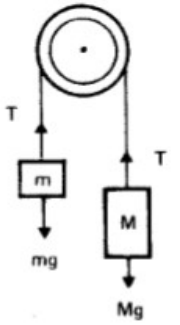
Experts' Explanations for Their Similarity Groupings

Expert 2: "*Conservation of Energy*"

Expert 3: "*Work-Energy Theorem*. They are all straight-forward problems."

Expert 4: "These can be done from energy considerations. Either you should know the *Principle of Conservation of Energy*, or work is lost somewhere."

Problem 5 (39)

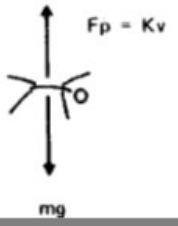


Expert 2: "These can be solved by *Newton's Second Law*"

Expert 3: "F = ma; *Newton's Second Law*"

Expert 4: "Largely use F = ma; *Newton's Second Law*"

Problem 12 (23)



Expertise in Problem Solving

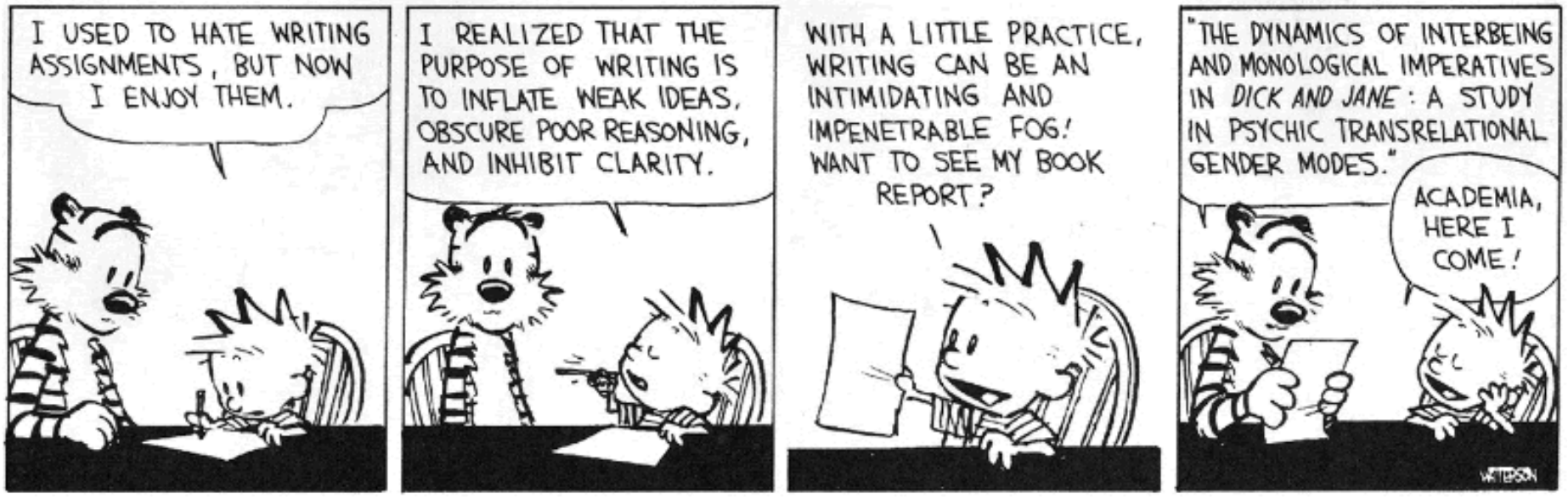
- "In this study, we specially designed a set of 20 problems to test the hypothesis that novices are more dependent on surface features, whereas experts focus more on the underlying principles. … We were able to replicate the initial findings that experts categorize problems by physical laws, whereas novices categorize problems by the literal components."

Expertise in Problem Solving

- "If we assume that such categories reflect knowledge schemata, then our results from the person at the intermediate skill level suggest that, with learning, there is a gradual shift in organization of knowledge --- from one centering on the physical components, to one where there is a combined reliance on the physical components and the physics laws, and finally, to one primarily unrelated to the physical components."

Expertise in Problem Solving

- "Improved ability to learn would be developed through a knowledge strategy in which individuals would be taught ways in which their available knowledge can be recognized and manipulated."
  - Do we do this in school?



Expert Bodies, Expert Minds

- U. Debarnot, M. Sperduti, F. Di Rienzo, and A Guillot. *Experts bodies, experts minds: How physical and mental training shape the brain*. Frontiers in Human Neuroscience, 2014.

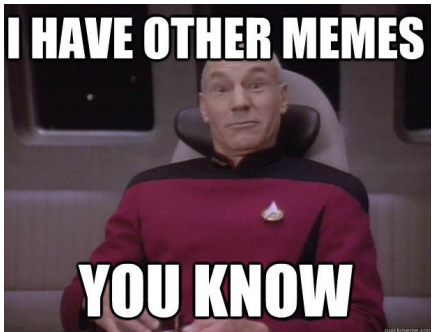
- Summary?

Expert Bodies, Expert Minds

- “These results suggest that the disparity between the quality of the performance of novice and expert golfers lies at the level of the functional organization of neural networks during motor planning. More generally, Patel et al. (2013) demonstrated that spatially distributed cortical networks and subcortical striatal regions may serve as neural markers of practice interventions.”
  - What's a “practice intervention”?

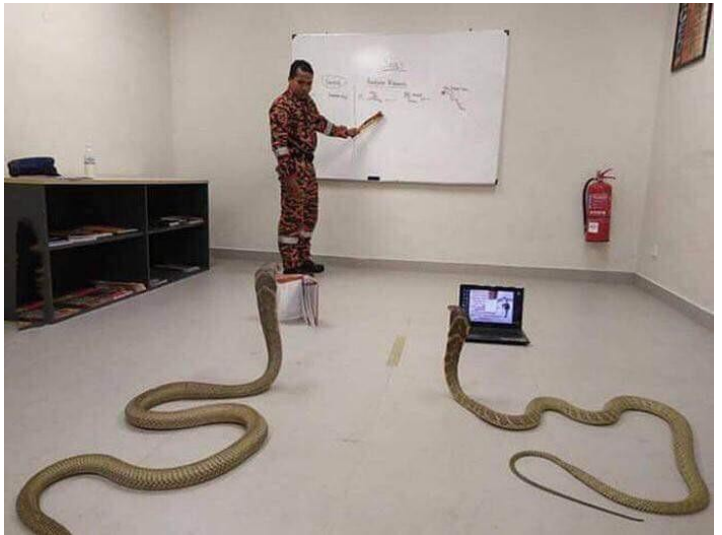
Expert Bodies, Expert Minds

- “Recently, Picard et al. (2013) examined the consequence of practice-dependent motor learning on the metabolic and neural activity in M1 of monkeys who had extensive training (~1–6 years) on sequential movement tasks. They found that practicing a skilled movement and the development of expertise lead to lower M1 metabolic activity, without a concomitant reduction in neuron activity. In other terms, they showed that less synaptic activity was required to generate a given amount of neuronal activity.”
  - What does this mean?



Expert Bodies, Expert Minds

- Scholz et al. (2009) reported experience-induced changes in white matter architecture following a short period of practice. Practically, it was found that 6 weeks of juggling practice protracted an increased fractional anisotropy in a region of white matter underlying the intraparietal sulcus.



Taxi Cab Drivers

- If the brain anatomy parts are a bit opaque, it may be easier to interpret a famous study of London taxi cab driver brains [http://www.scientificamerican.com/article/london-taxi-memory/]. Memorizing and navigating that spatial problem (London is not laid out on a clean grid) causes growth in the hippocampus. Quote:
  - “These navigational demands stimulate brain development, concludes a study five years in the making. With the new research, scientists can definitively say that London taxi drivers not only have larger-than-average memory centers in their brains, but also that their intensive training is responsible for the growth.”

Back To The Timed Exercise

- What are other ways to solve this?
  - Hint: I did not “write a program” at all in the conventional sense.
- If this were a contest (*and it is not!*), the key decision/mistake happened in the first seconds when you decided to write a program.
  - “C vs. Python” is a red herring: to phrase things as pejoratively as possible, that determines the winner of the loser's bracket.

What Predicts Software Developers' Productivity?

- E. Murphy-Hill, C. Jaspan, C. Sadowski, D. C. Shepherd, M. Phillips, C. Winter, A. K. Dolan, E. K. Smith, M. A. Jorde. *What Predicts Software Developers' Productivity?* Transactions on Software Engineering, 2019.
- " … a survey that asked 622 developers across 3 companies [Google, ABB, National Instruments] about these productivity factors and about self-rated productivity"

Self-Reported?

- "I regularly reach a high level of productivity."
- Correlate with some objective measures at Google (n=3344)
  - Senior devs self-report less productivity

Model	Factor	Estimate	Sig.	R^2
1	log(lines_changed + 1)	0.045	***	0.017
	level	-0.051	**	
2	log(changelists_created + 1)	0.112	***	0.024
	level	-0.050	**	
3	log(lines_changed + 1)	-0.015	n.s.	0.024
	log(changelists_created + 1)	0.132	***	
	level	-0.051	**	

The Results

bottom, the weakest. To determine which of the factors we can have the most confidence in, we identify the results that are statistically significant across all three companies:

- Job enthusiasm (F1)
- Peer support for new ideas (F2)
- Useful feedback about job performance (F11)

Discussion. A notable outcome of the ranking is that the top 10 productivity factors are non-technical. This is somewhat surprising, given that most software engineering research tends to focus on technical aspects of software engineering, in

- They also included COCOMO factors (what are those again?) and found that they didn't matter
  - Either COCOMO isn't accurate
  - Or it's accurate at the project, not person, level

Hypotheses

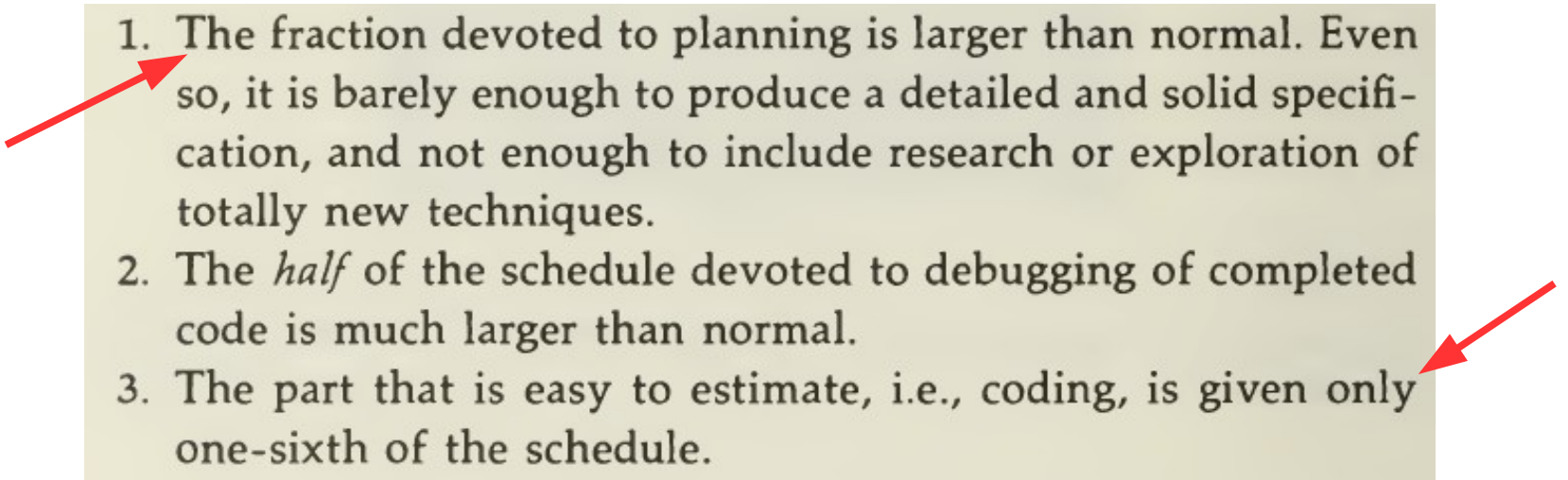
- ~~My computer is slow.~~
- I'm slow ~~and so is my program.~~
- I picked the wrong ~~language~~/abstraction and couldn't break up the problem.
- I did not recognize the true components of the problem.
- My brain is currently inefficient, requiring much metabolism for little neural activation.
- Non-technical factors, like peer support and feedback, correlate with productivity.

My Opinion: Programming Performance

- A substantial performance factor designated as "programming speed," associated with faster coding and debugging, less CPU time, and the use of a higher order language.
  - Programming Speed = Common Mistaken Belief!
  - Use of Abstraction = The Real Deal
    - The language is just one way to get abstraction. Abstraction (so that you can break up the problem and re-use existing solutions) is the relevant insight.

My Opinion: Mythical M-M

- "Planning" includes deciding whether write a standard program or whether to try something different ("totally new techniques")
  - Coding is much less relevant than many think.



1. The fraction devoted to planning is larger than normal. Even so, it is barely enough to produce a detailed and solid specification, and not enough to include research or exploration of totally new techniques.
2. The *half* of the schedule devoted to debugging of completed code is much larger than normal.
3. The part that is easy to estimate, i.e., coding, is given only one-sixth of the schedule.

My Opinion: Mythical M-M

- “The real insight is the observation of **language invariance**.
  - You can get 10 lines of ASM or 10 lines of Python.”
- All keystrokes in my solution to this problem
  - [Ctrl]-A cat > foo [Enter] [Ctrl]-V [Ctrl]-D vim foo [Enter] Vjjjjjjjjjd :%s/$/+/g [Enter] :0VGJA0 [Enter] V!bc -l [Enter] A/10000 V!bc -l [Enter]
- You can solve this by typing less, not faster.
  - Would typing 100% faster or slower have mattered?

My Opinion: Expertise in Problem Solving

- “Another interesting aspect of novice problem solving is not only that they commit more errors than experts but that, even when they do solve a physics problem correctly, their approach is quite different.”
- Story time: “I've seen this one before.”
  - Linux OOM Killer.
- “approach is quite different” cf. “new techniques”
  - Is “calculate math” a primitive in your language?

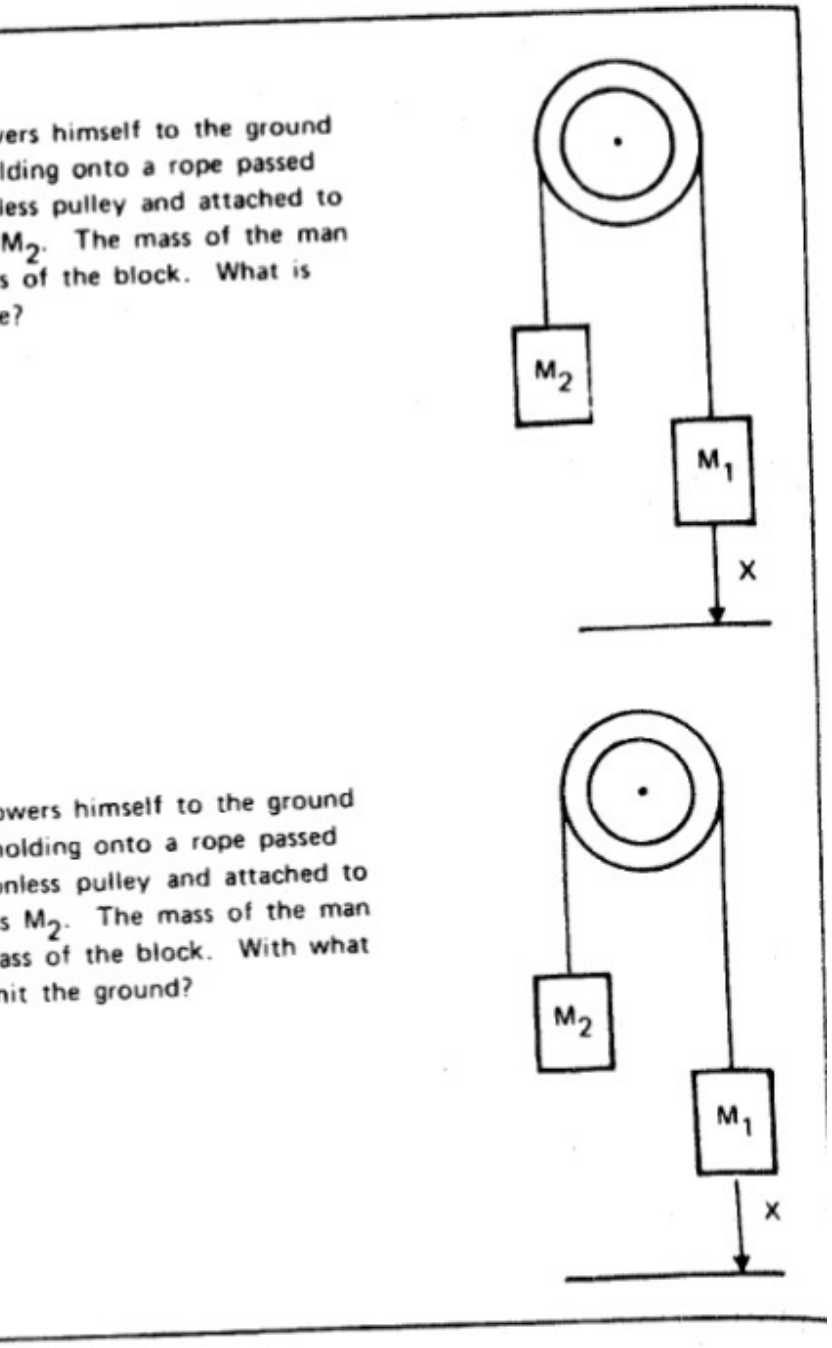


FIG. 1.6. Sample problems.

My Opinion: Problem Solving

- Many of you looked at the problem and, despite the instructions, saw that it looked similar to programming tasks you'd been given before.
- Those are “it looks like a pulley” surface features (file access then loop to compute total then divide).
- You wanted “it uses Newton's 2nd Law” deep features (compute the average).

My Opinion: Expert Bodies, Expert Minds

- On Page 6 (= Page 17) the Chi reading talks about three quantifiable (!) differences between experts and novices when solving problems.
  - The first is raw solution time (which we already saw in the Sackman reading).
  - The second is pauses in retrieving chunks of the correct equation. This is more interesting (cf. "chunking"): "experts group their equations in chunks so that the eliciting of one equation perhaps activates another related equation, and thus it can be retrieved faster". For programming, replace "equation" with "program fragment".
- One difference that previous students noted after watching my "how I did it" explanation was that I never really seemed to stop and think about what to do next, whereas a student might write the code to read in lines, stop and think, write the code to iterate over them and sum them, stop and think, write the print-and-divide code, etc. If you've observed that in yourself, the psych research summarized in the Chi reading suggests that one area for improvement is to get better at chaining from one fragment to the next.

My Opinion

- My “plan” breakdown:
  - This problem is regular expressions plus a calculator.
    - Use regular expressions to turn the input into an arithmetic expression (“into a program”)
    - Feed that to a pre-existing calculator
- Students who said “I will pass this to Excel” also did well.
  - Why are you re-inventing the wheel? Your boss wanted the right answer as fast as possible.

Advice 1/3: Small Potatoes

- Try to learn a shell-based editor, such as vim or emacs, and practice suspending the editor (ctrl-z, fg) rather than restarting it. If you must use something like Eclipse for a project, start it once and never quit it.
- Inasmuch as extra hand actions on your part are isomorphic to the computer delaying before giving you what you really want, master "focus follows mouse" (yes, even Windows supports it) and editors that don't involve new windows. Similarly, master keyboard shortcuts and favor an editor that allows you to make your own macros. Memorize the common ones shared across many interfaces, like ctrl-a (beginning of line) and ctrl-e (end of line -- those both work in the shell as well).
- Buy fast storage.

Advice 2/3

- Students often overemphasize the effect of low-level notions like typing speed but underemphasize high-level decisions (like breaking down a problem so its components can be solved in terms of transformations on existing solutions). When adding numbers, we demonstrated this concretely by taking what was to some a unitary atomic problem ("sum a list of numbers") into smaller parts ("turn a list of numbers into an arithmetic expression with regular expressions" and "invoke a calculator").
- This is non-obvious for a few reasons, not the least of which is that the parts actually appear to be larger, not smaller! So one trick is to gain enough felicity with various small problems in computer science that you can solve them quickly (see Sackman reading), as well as to retrieve them quickly and do the chunking to break down the big problem in terms of those parts (see Chi reading) without your machine setup actually getting in the way (see Dougherty reading).

Advice 3/3

- Ultimately, the bottleneck productivity limitation is not your typing speed. The real obstacle is more a conceptual limitation related to abstraction -- and there may be no shortcut to years of practice, the sort of study that ultimately changes the organization of your brain.
- Good luck.

Questions?

- Focus on HW6