We Mourn Our Craft — Sample

Swanand Kriyaban

Contents

Dedication

This work is offered with profound gratitude and humble salutations at the lotus feet of my Sadguru, Yogiraj Dr Mangeshda, my Spiritual Preceptor and guiding light.

Whatever is true in these pages flows from His grace. Whatever falls short is mine alone.

To Shambhavi — my wife, my first philosopher, my most persistent advocate. For years you have been telling me to write a book. For years I have been nodding and not writing one. You did not know this book existed. Surprise. Every late-night conversation we have had about life, about what matters, about what it means to make something with your hands and your attention — those conversations are the bedrock beneath every page. You were right. You are usually right. I should have started sooner, but then again, the dosa batter ferments on its own schedule.

Chapter 1: Fortran Is Looking Droopy

The cactus was dying, which was supposed to be impossible.

Karthik Sundaram had owned the small barrel cactus for ten years, through three office relocations, one company acquisition, and the mass extinction event that was the Cadence Systems renovation of 2024, during which every desk had been moved, every monitor recalibrated, and every plant temporarily exiled to a conference room that smelled of paint and broken promises. He had named the cactus Fortran, because like the language, it was old, resilient, and nobody under thirty understood why it was still around. Fortran sat on the corner of his desk in a clay pot that he'd brought back from a trip to Pondicherry — unglazed, slightly lopsided, with a thumbprint pressed into the base by whoever had made it — and for ten years it had asked nothing of him except to be left alone and occasionally watered.

Now it was turning yellow at the base.

"You're overwatering it," said Sanjay Mehta, appearing at the edge of his desk with two cups of coffee, one of which he placed near Karthik's keyboard with the gentle precision of a man defusing a bomb. Sanjay brought him coffee every morning at 9:15. It was one of the few remaining rituals at Cadence that had not yet been automated, optimized, or subjected to an AI-driven efficiency review.

"I haven't changed anything about how I water it."

"Then it's a sympathy death. It can sense the vibe in here. Even the plants are updating their résumés."

Karthik smiled. Outside the sixth-floor windows, February was doing its usual number on Cambridge — a colorless sky pressed down on the Kendall Square biotech buildings like a lid on a pressure cooker, and a wind off the Charles was finding creative routes through the building's HVAC system. Karthik had lived in Boston for twenty-one years now, and he still found the winters personally offensive, as if the weather had read his birthplace — Chennai, where February meant eighty-five degrees and jasmine — and decided to make a point.

"David sent another email," Sanjay said.

"I know."

"Did you read it?"

"I read the subject line. 'Exciting Update on Project Lighthouse.' That was sufficient."

"You should read the body. There's a paragraph where he uses the word 'synergy' and the phrase 'human-AI symbiosis' in the same sentence. It's beautiful, Karthik. It's like watching a man try to juggle buzzwords and accidentally set himself on fire."

Karthik turned from Fortran's declining health to his monitor, where thirty-four unread messages waited in Slack like small animals demanding to be fed. He was a Senior Staff Engineer, which meant he was technically one of the most experienced programmers at Cadence, and practically a person who spent sixty percent of his day in meetings, twenty percent answering Slack messages, and the remaining twenty percent doing the work he'd actually been hired to do, which was writing code. Good code. Code that did what it was supposed to do because he understood, at every level, what it was doing and why.

He had been doing this for twenty-one years. He was, by any reasonable measure, exceptional at it. The kind of engineer people called at 2 AM when something was on fire and nobody could find the accelerant. He could hold an entire system in his head the way he could hold a raga — all the notes, all the intervals, all the places where the melody could bend and where it could not — and when something was wrong, he could hear it. Not see it, not deduce it. *Hear* it. The way a musician heard a flat note in a chord. The way his mother, three thousand miles away in Adyar, could taste the missing ingredient in a sambar before the first spoonful had fully registered.

That kind of knowing was not something you learned from a tutorial. It was something that accumulated, slowly, over years, like silt in a river — each bug fixed, each system debugged, each late night spent tracing a memory leak through seven layers of abstraction. It was, Karthik thought, a form of *tapas* — not the Spanish appetizers but the Sanskrit kind. Heat. Discipline. The patient burning that transformed raw effort into understanding.

And lately he had the crawling feeling that this kind of knowing was becoming obsolete. Not wrong. Not inefficient. Just… unnecessary. Like being a master calligrapher in the age of the printer. Your hand was still steady, your eye was still true, but the world had decided that steady hands and true eyes were no longer the point.

He opened David's email.

The email was seven paragraphs long, which in corporate communication was the equivalent of a Tolstoy novel. David Tan, Vice President of Engineering, had a gift for writing sentences that contained many words and very little meaning, like a Rube Goldberg machine designed to transport air from one side of a room to the other.

Team,

I'm thrilled to share the next phase of Project Lighthouse, our initiative to position Cadence at the forefront of AI-native software development. After months of planning and collaboration with our leadership team, I'm excited to announce that beginning March 1, we will be transitioning to an AI-first engineering model across all product teams.

Karthik felt his stomach do something his yoga practice had not prepared him for.

What does this mean in practice? It means we're embracing a future where AI isn't just a tool in the engineer's toolkit — it's the primary author of our codebase. Our brilliant engineering team will evolve into a role that's even more impactful: guiding, reviewing, and orchestrating AI-generated solutions rather than writing code line by line.

"Orchestrating," Karthik said aloud.

"Wait for it," Sanjay said. He had pulled up a chair.

This transition will, naturally, require us to thoughtfully right-size our engineering organization to reflect these new efficiencies.

"There it is," Sanjay said. "'Right-size.' The word that means 'layoffs' but sounds like something you'd do to a pair of trousers."

Karthik kept reading, but the rest was the usual padding — mentions of generous severance packages, "transition support," an "incredible journey" that they had all been on together. David always talked about incredible journeys. Karthik had once told Sanjay that David described every corporate initiative like it was a Lord of the Rings movie, and Sanjay had replied, "Except in his version, Frodo would've outsourced the ring-carrying to an AI and then laid off Samwise for redundancy."

He closed the email and looked at Fortran. The cactus looked back, yellowing.

"How many?" he asked.

"The rumor is forty percent of engineering. Maybe more."

"Forty percent."

"Over three rounds. Through the end of the year. They're calling it 'phased transition' because 'slow-motion firing squad' didn't test well in the focus groups."

Karthik had been through layoffs before. Everyone in tech had, the way everyone in Boston had been through a nor'easter — you knew it was coming, you stocked up, and it still somehow shocked you when the wind hit. But this was different. Previous layoffs had been about money, about markets, about the cyclical nature of an industry that ran on hype and quarterly earnings. This was about something else. This was about the work itself becoming unnecessary.

Not inefficient. Not expensive. *Unnecessary*.

"Are you worried?" he asked Sanjay.

"About being laid off? No. I've been here nine years, I know where all the bodies are buried." He paused. "I mean that metaphorically. Although the codebase for the billing module is essentially a mass grave of good intentions, so maybe also literally."

"What are you worried about, then?"

Sanjay looked at his coffee. For a moment, the mask slipped — the joke-a-minute deflection he wore like armor — and Karthik saw something underneath that he recognized because he felt it too. Something that wasn't fear of unemployment but something older and stranger: the feeling of watching something you loved become something you didn't recognize.

"I'm worried," Sanjay said carefully, "that I'm going to spend the next six months supervising a machine that does my job, and that at the end of it, I'm going to realize that supervising isn't the part I liked."

The part Karthik liked — the part he had always liked — was the understanding.

He had tried to explain this to Meera once, on a Saturday morning in their kitchen, while he was grinding batter for dosa. The wet grinder hummed on the kitchen counter — an actual stone wet grinder, not a blender, which he'd ordered from a Tamil store in New Jersey and which Meera called "the loudest object in Massachusetts" — and Karthik was feeding soaked rice and urad dal into it in handfuls, watching the stones turn the grains into a smooth, pale paste.

"The thing about programming," he'd said, raising his voice over the grinder, "is that it's like this."

Meera had looked up from her laptop, where she was grading undergraduate papers on morphological analysis. "Like making dosa batter?"

"Like making anything where you understand every step. I know why the rice has to soak for six hours. I know why the ratio is three-to-one. I know that the fermentation tonight will depend on the temperature of the kitchen, and that in winter I need to leave the batter near the radiator, and that if I don't, the dosa tomorrow will be flat and pale instead of crisp and golden. I know all of this because I learned it — from my mother, from my own failures, from thirty years of paying attention. And because I know it, I can adapt. If the rice is a different variety, I adjust the water. If the kitchen is cold, I add a pinch of fenugreek. The knowledge isn't a recipe. It's a *relationship* with the material."

"And programming is like that?"

"Programming is exactly like that. You learn the machine. Not just what it does but *why* it does it. And once you understand the why, you can make it do things that nobody told you were possible, the same way a cook who understands fermentation can invent a dish that no recipe book contains."

Meera had smiled — the particular smile she reserved for moments when Karthik was being simultaneously correct and slightly ridiculous, which was, in her estimation, most of the time. "And the AI?"

"The AI is a machine that can produce perfect dosa without understanding fermentation. It can generate the batter instantly. The dosa will look right, taste right, fool anyone who eats it. But it doesn't *know* batter. It doesn't know why fenugreek helps in cold weather. And when something goes wrong — when the rice is different, when the temperature drops, when the conditions change in some way that the pattern doesn't account for — it won't adapt. It'll produce the same confident, perfect-looking dosa, and the dosa will be wrong, and nobody will know until they bite into it."

"That's very poetic for seven in the morning."

"I'm a poet. It's in the job description."

"You're an engineer who published one book of poems that sold forty copies."

"Forty-three."

"Forty-three copies. In Chennai. To people who were mostly related to you."

"My Aunt Kamala bought five. Gave one to her dentist."

Meera had laughed, and Karthik had laughed, and the grinder had hummed, and the morning had been good — the simple, earned goodness of a Saturday with the person you loved, in the house you'd made into a home, doing a thing you understood with your hands and your mind and your whole accumulated history of paying attention.

That was what he was afraid of losing. Not the job. The knowing.

At 10:30, Karthik had his first meeting of the day — a sync with his team about the migration of Cadence's notification service to the new AI-generated architecture. The meeting was in a glass-walled conference room called "Emerson," because Cadence had named all its conference

rooms after New England writers, a decision that led to sentences like "I'll be in Thoreau until two" and "Can you grab Hawthorne for the three o'clock?" and, on one memorable occasion, "We need to evacuate Dickinson, someone microwaved fish again."

Clara Santos was already there — his closest colleague, a fellow Senior Staff who'd been at Cadence even longer than he had. She'd brought her own coffee, which meant she'd already read David's email, which meant the look on her face was the look of a person who had processed bad news and arrived at the resigned-but-sharp stage of grief, the one where the jokes came out with edges.

"You read it," he said, sitting down.

"I read it. Did you notice he used 'incredible journey' twice? That's a new record. Usually he saves the second one for the all-hands."

"Sanjay thinks the sports metaphors are next."

" 'Stepping up to the plate' by Friday. 'Game plan' by Monday. 'Team players' — oh wait, he already used 'team players.' We're accelerating."

Priya Kapoor arrived then, sitting down with her laptop open and the specific anxiety of a junior engineer who'd been asked to present something and wasn't sure it was ready. Karthik had been watching Priya for months. She was bright — genuinely bright, with a quickness that reminded him of his best students at the MIT workshop he'd taught for two semesters. But she carried herself with the permanent slight alarm of someone who suspected the floor might disappear.

"How's the migration looking?" Karthik asked.

"It's — okay. I think. The AI generated the new service handlers over the weekend, and I've been reviewing them." She paused. "They work."

"But?"

"But I don't — I mean, they work, the tests pass, the integration tests pass, everything looks right. But there's this one section in the retry logic where it's doing something I don't fully understand. Like, structurally I can see what it's doing, but I can't tell you *why* it chose that particular approach. And when I asked the AI to explain, it gave me an explanation that was confident and detailed and might have been completely wrong."

Karthik nodded. He knew this feeling precisely. It was the feeling of listening to someone play a raga you knew well — the notes were correct, the rhythm was correct, but something was off. The *phrasing*. The intention behind the notes. The thing that turned sound into music was missing, and in its place was a perfect, soulless reproduction that passed every test except the one that mattered: did you understand what you were hearing?

"Show me," he said.

Priya turned her laptop around, and for the next twenty minutes they went through the retry logic together, line by line, the way Karthik had been taught at MIT and had taught himself in the decades since — the patient, deliberate practice of reading code the way you'd read a poem, attending not just to what the words did but to what they *meant*.

"This section here," he said, pointing. "The jitter calculation. It's using the failure count as a seed modifier. Do you see that?"

"I see it. I don't know why."

"Neither do I, yet. But I know how to find out. You trace it. Concrete values. Failure count one — what's the seed? What's the jitter range? Failure count two — how do they change? You follow the logic with your own mind, the way you'd follow a melody with your own ear. Not to verify that it's correct — to understand *why* it's shaped the way it is."

"That sounds like it would take a while."

"About forty minutes. Maybe fifty. And at the end, you'll know. Not 'the tests pass' know. *Know* know. The kind of knowing that lets you fix it at three AM when it breaks, because things you don't understand always break at three AM — it's a universal constant, like gravity, or the T running late."

Priya smiled at that, and Karthik felt the small, warm satisfaction of having said the right thing — the teacher's satisfaction, which was different from the engineer's satisfaction but related to it. Both were acts of transmission. Both required you to understand something well enough to make it visible to someone else.

Clara, who had been listening, caught his eye and gave him the faintest nod. She knew what he was doing. She did it too. They were both, in their way, trying to pass something down — a way of working, a way of thinking — before the river changed course and the knowledge was stranded on a bank that

nobody visited anymore.

Lunch was at Felipe's, the taqueria on Brattle Street that had been there since before Karthik moved to Cambridge and would, he suspected, survive whatever apocalypse eventually came for the rest of them. He went with Sanjay, because lunch with Sanjay was the closest thing to therapy that his insurance didn't cover, and also because Sanjay had an unmatched talent for making terrible situations funny, which was a form of kindness that Karthik valued above almost all others.

"I've been thinking about David's email," Sanjay said, unwrapping a burrito the size of a small infant.

"You've been thinking about it for three hours?"

"I've been thinking about it since the rumor started in November. But specifically, I've been thinking about the word 'orchestrating.' We're going to 'orchestrate' AI-generated solutions. Do you know what an orchestrator does?"

"Conducts an orchestra?"

"Exactly. An orchestrator doesn't play an instrument. An orchestrator stands in front of people who play instruments and waves a stick. David is telling us that our job is now to wave a stick."

"Some conductors are extraordinary."

"Sure. Zubin Mehta was extraordinary. But Zubin Mehta understood music. He could sit at a piano and play anything. He could hear a wrong note in a hundred-piece orchestra and identify which second violinist was a half-step flat. That's what made him a great conductor — not the waving, but the understanding." Sanjay took a bite of his burrito. "David wants us to wave the stick without learning the instruments."

Karthik thought about his bansuri, hanging in its cloth bag on the hook in the music room. The bansuri was, of all the instruments he'd tried over the years, the most honest. Six holes, a blowing hole, a piece of bamboo. No keys, no valves, no mechanism between your breath and the sound. Everything depended on you — the angle of your lips, the pressure of your fingers, the shape of the air inside you. You couldn't fake it. You couldn't approximate. Either you understood the relationship between your body and the bamboo,

or you produced noise.

An AI could synthesize a perfect bansuri performance. Every note in tune, every ornament executed with mathematical precision. But it wouldn't be playing. It would be producing the artifact of playing without the act. And the difference — the difference that a listener might not consciously detect but that a player would feel in their bones — was understanding. Was *relationship*. Was the accumulated knowledge of ten thousand hours of sitting with a piece of bamboo and learning, breath by breath, what it could do.

"I ran the numbers on #things-the-ai-got-hilariously-wrong," Sanjay said, brightening. "We've had 247 submissions since October. The best one this week was from Chen Wei on the payments team. He asked the AI to refactor a date-handling function and it produced code that thought February had thirty days."

"Thirty?"

"Thirty. Not twenty-eight, not twenty-nine. Thirty. It just decided February was longer than it is. Like an optimist."

"Did it at least handle leap years?"

"Karthik, it gave February *thirty days*. The concept of leap years was several exits behind it on the highway."

Karthik laughed. He laughed because it was funny, and because laughing was easier than thinking about what it meant that the tool they were all about to depend on could confidently, fluently, persuasively produce code that got the number of days in February wrong. Not because it was stupid — it wasn't anything, it had no capacity for stupidity any more than a river had a capacity for rudeness — but because it didn't *know*. It didn't know that February had twenty-eight days the way Karthik knew it, the way his mother knew the exact moment to add tamarind to the rasam. It had a statistical relationship with the concept of February that was, most of the time, correct, and occasionally, confidently, wrong.

And there was no way to look at the code it produced and know which kind of February you were getting.

He drove home that evening — the Pike westbound from Cambridge, which

at 6 PM was less a highway than a parking lot with ambitions, a forty-mile procession of brake lights that moved with the speed and enthusiasm of a government bureaucracy. The commute was fifty minutes on a good day and ninety on a bad one, and today was a bad one, which meant Karthik sat in traffic past the Newton tolls with his hands on the wheel and his mind on David's email and a Hariprasad Chaurasia recording playing through the speakers, the bansuri notes filling the car with a beauty that the Massachusetts Turnpike did not deserve.

The house was on a quiet street in Westborough, a colonial with gray clapboard siding and a red front door that Meera had insisted on because she believed a front door should have a personality. They'd bought it in 2016 — four bedrooms, more space than two people strictly needed, but Karthik used one as a music room and Meera used one as a study, and the kitchen was big enough for the wet grinder and the stone mortar and the three different sizes of kadai and the cast-iron tawa and all the other equipment that South Indian cooking demanded and that Meera called "the arsenal." The backyard had a vegetable garden that produced tomatoes in summer and guilt in winter, when Karthik looked at the empty beds and felt he was failing the soil.

Meera was at the kitchen table when he came in, grading papers with a red pen in one hand and a cup of chai in the other, her reading glasses pushed down her nose in the way that made her look, Karthik thought, like a beautiful librarian who had been interrupted mid-discovery.

"How was it?" she asked, not looking up. This was one of the things he loved about her — she could read his mood from the sound of the front door and the weight of his footsteps in the mudroom. Heavy footsteps meant bad day, and bad day meant don't look up immediately, give him a moment to hang his coat and stand in the kitchen doorway and breathe before asking.

"David announced Project Lighthouse."

"The AI thing?"

"The AI thing."

Now she looked up. Her eyes, behind the glasses, were sharp and warm at the same time — the eyes of a woman who studied how language worked and could therefore detect the precise weight of what was being left unsaid.

"How bad?"

“Forty percent of engineering.”

“Forty percent.”

“Over three rounds. Through the end of the year.”

Meera took off her glasses and set down her pen. “Are you—”

“I’m safe. For now. Senior staff, institutional knowledge, blah blah. They need someone who can actually read the code when the AI writes something insane.”

“That’s a depressing reason to keep your job.”

“Everyone keeps saying that. Sanjay said it. Clara said it. It’s apparently the consensus view — ‘congratulations, you’re not fired, because you’re the last person who understands how the building is wired.’ ”

“It sounds like being the last doctor in a town that’s decided to replace medicine with crystals.”

Karthik laughed. Meera’s analogies were always better than his. It was one of the enduring small injustices of their marriage — he was the published poet, and she was the one who could nail a metaphor in conversation without pausing for breath.

He changed out of his work clothes, put on the soft cotton kurta he wore at home — white, with a thin blue border, the kind his father wore on every evening of Karthik’s childhood in Adyar, sitting on the veranda with the newspaper while the jasmine in the courtyard opened in the cooling air. He went to the kitchen window and checked his herbs. The curry leaf plant was holding on, which was a minor miracle; the mint was thriving because mint thrived everywhere, like a vegetable cockroach; the tulsi was looking peaked, possibly in sympathy with Fortran.

“I’m going to practice for a bit,” he told Meera.

“Of course.”

He took the bansuri from its hook in the music room — a bass bansuri in E, made of Assam bamboo, given to him by his guru in Chennai when he was nineteen. It was warm in his hands, the bamboo smooth from over two decades of handling. He sat on the cushion in the corner of the living room that was his practice space — a square of floor between the bookshelf and

the window that contained a zafu cushion, a small brass lamp, and nothing else. The simplicity was the point. When he sat here, there was him and the bamboo and the air, and nothing else was required.

He played. Not a formal raga — not tonight. Tonight he played long, slow notes, the kind that were more breath than sound, the kind that his guru called "the notes between the notes." He played the way you'd stretch after a long day of sitting — not to accomplish anything but to remind your body that it could move. The sound filled the house, low and warm, and Meera graded papers to the sound of it, and the curry leaf plant leaned toward the window, and the evening settled into the gentle, habitual peace of a life that was, in all the ways that mattered, good.

But under the peace — under the bansuri and the chai and the sloped floors and Meera's red pen — something had shifted. Something small. A hairline crack in the foundation that you might not notice if you weren't paying attention, but that Karthik noticed, because paying attention was the thing he was best at.

The crack was this: for the first time in twenty-one years, he was not sure that paying attention was enough.

He played a note and let it fade. In the silence after — that gap, that pause, the space his guru had taught him was where the real music lived — he thought about February, and thirty days, and the difference between knowing and not knowing. He thought about David Tan waving a stick. He thought about Priya, twenty-six years old, who had never known programming without AI the way a person born after the invention of the automobile had never known walking as the only way to travel.

He thought about his father, in the PWD office in Chennai, drawing bridge specifications by hand on drafting paper, every line a decision, every dimension a commitment, and how his father had once said, "Karthik, a bridge is a promise. You are promising the people who cross it that you understand every force that acts upon it. If you don't understand, you have no right to build."

He put the bansuri down and sat in the silence and wondered what it meant to build when the building was done by something that made no promises, because it couldn't, because it didn't know what a promise was.

Meera appeared in the doorway.

“Dinner?” she said. “I made rasam.”

“You made rasam on a weeknight?”

“You sounded like you needed rasam.”

This was love, Karthik thought. Not the grand, cinematic kind. The kind where someone listened to your footsteps and your bansuri and the weight of your silence, and made you rasam on a Tuesday.

“Coming,” he said. He stood, and put the bansuri back on its hook, and went to eat soup with his wife, and the crack in the foundation remained, small and quiet and patient, waiting.

Chapter 2: The Imposter's Paradox

Priya Kapoor had a theory about imposter syndrome, which was that it was significantly less distressing when you were confident it wasn't true.

The standard version went like this: you were good at your job, but you felt like a fraud. This was unpleasant but ultimately manageable, because underneath the anxiety was a floor — a bedrock of actual competence that you could touch if you reached far enough down. The panic was the surface weather. The skill was the geology. You might feel like an imposter, but you *weren't* one, and eventually a therapist or a friend or a sufficient accumulation of evidence would convince you of this, and you would go on feeling vaguely fraudulent but functional, which was, as far as Priya could tell, the baseline emotional state of most professionals.

Priya's version was different.

Priya's version was: what if you were an imposter?

Not metaphorically. Not in the "oh, everyone feels that way" sense that well-meaning people invoked at networking events while holding glasses of wine they'd gotten for free. But actually, genuinely, in a way that would be measurable if anyone thought to measure it. What if the reason she felt like she didn't know how to program was that she, in some fundamental and increasingly non-trivial sense, didn't know how to program?

She thought about this at 11:47 PM on a Tuesday night in her apartment in Allston, sitting cross-legged on her bed with her laptop open and a tab loaded with a blank file in her text editor. The apartment was a two-bedroom

that she shared with her roommate, Dani, who was a nurse at Mass General and was currently asleep with the disciplined efficiency of someone who had learned to fall unconscious in any position within thirty seconds. Dani could sleep through fire alarms. Priya had tested this theory accidentally when she'd burned a frozen pizza in October, and Dani had emerged from her room fourteen minutes after the smoke detector stopped, blinking, and asked if something smelled good.

The apartment was on Brighton Avenue, above a laundromat that ran its industrial dryers until midnight, which meant the floor vibrated gently for most of the evening, like living on the back of a large purring animal. The rent was $1,850 a month for Priya's share, which her parents in Edison, New Jersey, considered criminal and which Priya considered, by Boston standards, a stroke of miraculous luck. She had a window that faced the street, a closet that could fit approximately one and a half human bodies, and a shelf of vintage video game cartridges that she'd been collecting since college — mostly NES and SNES games that she'd bought at retro stores and flea markets and a memorable estate sale in Framingham where a ninety-year-old woman had sold her grandson's entire childhood for forty dollars.

Priya was terrible at all of them. She couldn't get past World 3 in Super Mario Bros. She had never once successfully landed the plane in Top Gun. She owned a copy of Battletoads, which was widely regarded as one of the hardest games ever made, and she had died on the first level so many times that she'd started to take it personally, as if the game had developed a specific animosity toward her.

She collected them anyway. She liked the cartridges themselves — the satisfying weight of them, the label art, the physical reality of a complete, finished program that fit in your hand. You could blow on the contacts and slide it into the console and it either worked or it didn't. There was no update, no patch, no server to authenticate with. The game was the game. It had been the game since 1987 and it would be the game forever. She found this deeply, almost spiritually, comforting, though she could not have articulated exactly why.

The blank file on her laptop was an exercise she'd been doing — secretly, the way other people might secretly eat fast food or watch reality television — for the past three weeks. The exercise was simple: write a program without AI assistance. Not a complex program. Not a production system. Just a small,

self-contained piece of code that did a specific thing, written entirely from her own knowledge, with nothing but the standard library documentation for reference.

Tonight's task: a function that took a list of integers and returned the second-largest unique value. That was it. A problem she could have found in any introductory programming textbook, the kind of thing that appeared in coding interviews as a warm-up, the appetizer before the real questions arrived.

She stared at the blank file.

She knew the shape of the solution. She could see it the way you could see the outline of a word you'd forgotten — the number of syllables, maybe the first letter, the feeling of it on the tip of your tongue. Sort the list. Remove duplicates. Return the second element. Or: iterate through, track the top two. She knew the general approach the way she knew the general layout of a city she'd visited once — confident about the major streets, uncertain about the turns.

She started typing.

```
def second_largest(numbers):
```

Good. Fine. A function definition. She could do function definitions. She had been writing function definitions for four years, since her sophomore year at Rutgers, when she'd taken Introduction to Computer Science and discovered that she liked programming the way some people liked crossword puzzles — the satisfaction of fitting things together, the click of a solution snapping into place. She'd switched her major from biology. Her mother had cried, not from disappointment but from confusion, because in her mother's world the career path went doctor, lawyer, engineer, accountant, and then a vast uncharted wilderness labeled "other" in which no one from Edison, New Jersey, had ever survived.

```
def second_largest(numbers):
    if len(numbers) < 2:
        return None
```

Edge case. Good. She felt a small flush of competence. She knew about edge cases without being told. That was something.

But now the middle part. The actual logic. She needed to — what? Sort? Use a set first to handle duplicates? Or track the two largest as she went?

Which was more efficient? Did it matter for this exercise? She could feel the phantom limb of the AI assistant, the tab-complete that wasn't there, the ghostly suggestion that would normally appear in gray text after she typed a few characters, offering to finish her thought before she'd fully had it.

She typed, deleted, typed again.

```
def second_largest(numbers):
    if len(numbers) < 2:
        return None
    unique = list(set(numbers))
    if len(unique) < 2:
        return None
    unique.sort(reverse=True)
    return unique[1]
```

She stared at it. It was seven lines. It was, she was fairly sure, correct. She could reason through it: convert to a set (remove duplicates), convert back to a list (so she could sort), sort descending, return the second element. It was not elegant. It was not clever. A more experienced programmer might have done it in a single pass through the list, tracking the top two values, which would have been O(n) instead of O(n log n) because of the sort, and she knew this — she knew this the way she knew that Battletoads had levels after the first one, theoretically, even though she'd never seen them.

But it was *hers*. She'd written it. Every character had come from her own fingers, informed by her own understanding, without an AI suggesting or completing or generating or hallucinating. It was small and it was clumsy and it was real.

She felt absurdly proud, and then immediately, crushingly embarrassed, because she was a professional software engineer at a major technology company and she was sitting in her bed at midnight feeling proud of herself for writing seven lines of code that solved a problem from an introductory textbook. This was like a professional chef being moved to tears by successfully boiling an egg.

She closed the laptop and lay back on her bed and stared at the ceiling, which had a water stain in the shape of Florida that she and Dani had named Gerald.

"Gerald," she said to the ceiling, "am I a fraud?"

Gerald, being a water stain, did not respond, which Priya appreciated. She was tired of things that responded confidently.

She had gotten the job at Cadence eighteen months ago, which was six months after graduating from Rutgers with a degree in computer science and a GPA that was respectable without being remarkable, like a sedan — it would get you where you needed to go but nobody was going to photograph it for a magazine. She'd interviewed at fourteen companies, which was fewer than most of her classmates (the average was something like twenty-three, a number that made the whole enterprise feel less like a job search and more like a season of a dating show where the prize was health insurance).

The Cadence interview had gone well, mostly because the technical portion had been conducted with AI tools available, which was how most companies did it now. They gave you a problem, you solved it using whatever tools you normally used, and they evaluated your approach, your questions, your ability to verify the AI's output and catch its mistakes. Priya had caught two bugs in the AI-generated solution, which impressed the interviewer, a kind-eyed woman named Reshma who had said, "Good eye — most candidates miss the off-by-one error."

Priya had not told Reshma that she'd spotted it only because she'd made the same off-by-one error herself so many times that she'd developed a kind of scar-tissue intuition for it. She'd wanted to seem talented rather than traumatized.

The job itself was — fine. Good, even. She worked on the notification service, which was the system that sent emails, push notifications, and in-app messages to Cadence's customers. It was not glamorous work. Nobody had ever said, "I got into computer science because I dreamed of one day sending millions of push notifications about features nobody asked for." But it was real work, with real users, and Priya took it seriously.

Her process, which she had never described aloud to anyone because she suspected it would sound insane, was this:

1. Read the ticket. Understand what was being asked.
2. Think about the problem. Sketch the approach in her head or on paper.
3. Open the AI assistant. Describe the problem. Let it generate a solution.
4. Read the AI's solution carefully, comparing it to what she'd expected.

5. If the solution matched her mental model, great — she'd clean it up, test it, submit it.
6. If the solution didn't match her mental model, she'd try to figure out whether the AI was smarter than her or wrong. This was the hard part, because the AI was frequently smarter than her *and* wrong at the same time — it could produce code that was sophisticated and broken in a way that was sophisticated and broken differently from how she would have been sophisticated and broken.

The problem — the thing that kept her awake at night and drove her to write seven-line programs in bed like a secret drinker hiding bottles — was step 2. The thinking. The mental model. She suspected that her mental model was getting thinner. Not wrong, exactly, but… shallow. Like a riverbed that was slowly drying out because the water had been diverted somewhere else. She could still think about problems, but the thinking felt effortful in a way that she worried was atrophy rather than rigor.

She had started learning to program in 2021, as a sophomore at Rutgers — before the AI tools arrived. For one blessed year, she had written code the old way: staring at a blank editor, Googling error messages, reading Stack Overflow threads from 2014 that began with "This question has been marked as duplicate" and ended with an answer that was either exactly what she needed or completely wrong, with no reliable way to tell the difference. Then, in her junior year, Copilot arrived. Then ChatGPT. Then everything else, in a rush, like a dam breaking. By her senior year, AI tools were not just available but *expected* — woven into the IDE, integrated into the workflow, offered by professors who'd spent the previous summer rewriting their syllabi in a state of controlled panic. Her professors at Rutgers had been divided — some banned AI tools entirely, which felt like banning calculators in a math class; some embraced them wholesale, which felt like letting students bring a Michelin-starred chef to a cooking exam. The ones Priya had liked best were the pragmatists who'd said, essentially, "These tools exist, you will use them, but you need to understand what they're doing, and the only way to understand what they're doing is to first understand how to do it yourself."

The problem was "first." First implied a sequence — learn it yourself, then use the tools. But Priya had barely survived one year of learning-it-herself before the tools showed up and offered to carry the weight. And she'd let them. Of course she had. Who wouldn't? And struggling, she was start-

ing to suspect, was the part that mattered. Struggling was how the knowledge got *in* — not just into your short-term memory, where it could answer a quiz question, but into the deep architecture of your thinking, where it became intuition. Where it became the thing that let Clara look at a piece of retry logic and say, "I don't understand why it's doing this," and have that not-understanding mean something, because she had a framework of understanding against which the absence registered.

Priya didn't have the framework. She had the tools. And she was beginning to worry that having the tools without the framework was like having a GPS without knowing how to read a map — fine as long as the GPS worked, catastrophic the moment it didn't.

Wednesday morning she got to the office early, which at Cadence meant 8:45, because the engineering culture was nominally "flexible" but in practice revolved around the unspoken understanding that if you weren't at your desk by 9:00, someone would assume you were either slacking or dead, and would investigate with equal concern for both possibilities.

The sixth floor was quiet. The open plan, which during peak hours resembled a nature documentary about a particularly anxious colony of meerkats — heads popping up over monitors, swiveling toward sounds, ducking back down — was temporarily peaceful. Priya made herself a coffee from the machine in the kitchen, which was a Jura E8 that had cost more than her first car and produced espresso that tasted like someone had described espresso to a machine that had never experienced joy. She added milk and sugar in quantities that a coffee purist would consider criminal, carried it to her desk, and opened her laptop.

Thirty-one Slack messages. Fourteen emails. Two pull request reviews. A calendar invitation for a meeting called "Lighthouse Readiness Check" that had been scheduled by David Tan's executive assistant with the ominous brevity of a jury summons.

She opened the retry logic from yesterday — the code she'd shown Karthik. It was still there on her screen, still doing the thing she didn't understand. She read through it again, slowly, the way Karthik had read through it with her, line by line.

The AI had generated a retry mechanism that used an exponential backoff

with jitter, which was standard. But embedded in the jitter calculation was a conditional that adjusted the randomization seed based on the number of previous failures, in a way that created a pattern that was deterministic-looking but wasn't quite. It was the "wasn't quite" that bothered Priya. She could see that it *worked* — the tests proved it worked — but she couldn't see *through* it. She couldn't follow the thread from intention to implementation the way she imagined Clara could.

She'd asked the AI to explain it. The AI had said:

This approach implements an adaptive jitter strategy that correlates the randomization window with the failure count, creating a pseudo-deterministic retry pattern that reduces thundering herd effects in distributed systems while maintaining sufficient entropy to prevent synchronized retries across service instances.

Which sounded authoritative and might have been completely made up. Priya had no way to distinguish a correct explanation from a confidently wrong one, because she didn't have the underlying knowledge to check it against. It was like asking someone for directions in a language you didn't speak — the tone could be helpful or mocking or completely fabricated, and you'd smile and nod either way.

She was still staring at it when Karthik arrived at 9:10, unwinding a wool scarf and carrying the faint, warm smell of cardamom that always seemed to follow him, as if the spices from his morning kitchen had stowed away in the fibers of his coat for the commute from Westborough.

"You're early," Karthik said.

"I'm looking at the retry logic again."

Karthik set down his bag and came to look over Priya's shoulder. He studied the screen for a moment with the particular stillness he brought to reading code — the same stillness, Priya imagined, that he brought to his flute, or his yoga, or whatever other practice required you to be quiet long enough to hear what was actually there.

"I think," Karthik said after a moment, "it's implementing a version of decorrelated jitter. It's a known pattern. But the way it's doing it is unusual — it's folding in the failure count as a seed modifier, which I've seen in some distributed systems papers but not in production code. It might be correct and

clever, or it might be a hallucinated hybrid of two different approaches."

"How do you tell the difference?"

"The same thing I showed you yesterday. Concrete values. Sit with it. Pick a failure count, calculate the seed, work out the jitter range. Then do the next one. Follow the logic wherever it goes — the way you'd follow a melody, note by note, until you hear the shape of it."

"That sounds like it would take hours."

"About forty minutes. And at the end of it, you'd know. Not 'the tests pass' know. *Know* know."

"Or I could run the fuzz tests and see if it breaks."

Karthik looked at her. It wasn't a judgmental look — Karthik was not, as far as Priya could tell, capable of the kind of performative disappointment that some senior engineers wielded like a weapon. It was a look of gentle recognition. Like he saw something in Priya's suggestion that confirmed something he'd already been thinking.

"You could," Karthik said. "And if the fuzz tests pass, you'd know it works. But you still wouldn't know *why* it works. It's like — you know when you taste a dish and you can tell it's good, but you can't tell me what spice is making it good? That's knowing-by-result. Now imagine the dish goes wrong one day and you have to fix it, but you never learned what was in it. That's what happens when code you don't understand breaks at three AM on a Saturday — and things you don't understand always break at three AM on a Saturday, it's a universal constant, like gravity, or the T running late."

"So I should spend the forty minutes."

"I think you should spend the forty minutes."

Priya nodded. She turned back to her screen and felt, simultaneously, two contradictory things: gratitude that someone was telling her to do the hard thing, and resentment that the hard thing was necessary when the easy thing was *right there*, and the easy thing worked, and no one except Karthik seemed to think the distinction mattered.

At lunch, Priya ate at her desk — a grain bowl from Sweetgreen that cost four-

teen dollars and contained enough kale to feed a small horse but not quite enough calories to feed a twenty-six-year-old human through a full workday. She scrolled through her phone while eating, a habit that she knew was "bad for mindfulness" and that she had absolutely no intention of changing, because mindfulness, in her experience, was the practice of being fully present with your anxiety, which she could do just fine without a meditation app.

Her college friend Kenji had texted a link to the original blog post that everyone at work was apparently reading — the one called "We Mourn Our Craft." She opened it, skimmed it, and felt the specific discomfort of reading something that described your private fear in someone else's words.

The post was by a programmer who had been writing code for years and was watching AI transform the profession. The central argument was that programming was a craft — not just a job, not just a skill, but a way of thinking, a way of relating to the world — and that this craft was being hollowed out by tools that could produce the artifacts of programming (the code, the tests, the documentation) without the understanding that had always been the craft's beating heart.

Priya read it and felt two things.

The first was sympathy. She understood the grief. She could see it in Clara every day — the way Clara read code like a novel, savoring the structure, noticing the choices, understanding the *intent* behind every line. It was beautiful. It was like watching a master carpenter run her hands over a joint and know, from touch alone, whether it was true.

The second thing Priya felt was something she was less proud of, something she would not have said aloud, but which she thought in the privacy of her own skull with the honesty that only Gerald the water stain was privy to: *I never had the craft to mourn.*

That was the thing. The thing that separated her from Clara and Sanjay and Mervin Webb and all the other programmers who wrote blog posts about the death of their profession and debated it on Hacker News with the passionate intensity of people arguing about something they loved. They had had the thing, and they were losing it, and the losing was painful. Priya had never had it. She had arrived at the party after the band had stopped playing, and everyone was sitting around talking about how incredible the music had been, and she was standing in the doorway holding a bottle of wine she'd

brought and wondering if she'd missed the whole point.

She couldn't mourn what she'd never had. But she could — and this was worse, she thought, this was actually worse — she could mourn the fact that she'd never had it. She could grieve the absence of the thing, the knowledge that there was a depth of understanding she'd never reached because the tools had always been there to catch her before she fell far enough to learn how to land.

Her phone buzzed. Kenji again.

Did you read it?

Yeah.

Thoughts?

She typed and deleted three responses before settling on:

I think it's written by someone who learned to swim before they invented boats, and they're upset that people are using boats now. And they're not wrong that something is lost. But also — some of us never learned to swim.

Kenji sent back a thinking face emoji, which was his way of saying he didn't know how to respond but wanted her to know he'd read it.

Priya put her phone down and looked at the retry logic on her screen. She opened a new document and started working through the math by hand, the way Karthik had suggested. Failure count one. Seed value. Jitter range.

It took her fifty-three minutes, not forty. At the end of it, she understood what the code was doing. It was, in fact, implementing decorrelated jitter with a failure-weighted seed — and it was doing it correctly, with one small exception: at very high failure counts (above twenty), the seed modifier would overflow a 32-bit integer boundary, which would cause the jitter to collapse to zero, which would cause all retries to synchronize, which would cause exactly the thundering herd effect the jitter was designed to prevent.

The AI had written almost-correct code. The kind of code that would pass every test, survive every fuzz run, work flawlessly in production for months or years — until the day a service degradation lasted long enough to push the failure count above twenty, at which point the retry logic would turn into a synchronized DDoS attack on Cadence's own infrastructure.

Priya stared at the bug. She understood it. Not because the AI had told her, not because a test had caught it, but because she'd spent fifty-three minutes tracing the logic with her own mind, and her own mind had followed the thread to the place where it frayed.

She felt something she couldn't name. Something that was adjacent to pride but deeper — more architectural. Like a beam had been placed in a structure she hadn't known she was building.

She Slacked Karthik.

Found a bug in the retry logic. Integer overflow at high failure counts. Jitter collapses to zero above ~20 retries.

The response came ninety seconds later.

Good. That's a real bug. It would have gone to production.

Then, a moment later:

That's what understanding is for.

Priya looked at the message. She looked at the code. She looked at Gerald.

"Okay," she said quietly. "Fifty-three minutes."

She could do fifty-three minutes.

That night, back in the apartment, the dryers humming beneath her floor, Dani asleep already at 9:30 because her shift started at 5 AM, Priya sat on her bed with her laptop and the blank file open.

Tonight's exercise: a function that determined if a string of parentheses was balanced. Another textbook problem. Another thing that the AI could generate in two seconds flat, perfect and gleaming and entirely opaque.

She started typing.

It took her twelve minutes. She made two mistakes, both of which she caught. The final version was fourteen lines long and used a counter instead of a stack, which was technically less general — it wouldn't handle multiple types of brackets — but for the specific problem of matching parentheses, it was correct and she understood every line.

She saved the file. She closed the laptop. She picked up a copy of Mega Man 2 from her shelf — the cartridge heavy and solid in her hand, the label art showing the Blue Bomber in mid-jump, frozen forever in a moment of determined, pixelated action.

She thought about Clara's father, coming home to find his ten-year-old daughter's name scrolling across a TK-85 screen in their apartment in São Paulo. She thought about the feeling that Clara had described — the magic of telling a machine what to do and having it listen.

She hadn't felt that yet. Not fully. But tonight, for fifty-three minutes, she'd felt something that might have been its beginning. A small, hard, earned thing, like a seed. Like a piece of understanding that hadn't been given to her or generated for her but that she'd found on her own, by reaching into the code and following the thread to the place where it broke.

She put the cartridge back on the shelf and turned off the light.

The dryers hummed. Gerald watched from the ceiling. Somewhere in Allston, a siren wailed, and somewhere else, a train rattled past, and somewhere in the vast humming network of servers that kept the modern world running, a piece of code that nobody fully understood continued to work correctly, for now, while it waited for the twenty-first failure.

Chapter 3: The Last Comprehensible System

Mervin Webb had not set out to build a monument. He'd set out to build a tool.

The distinction mattered to him, though he was aware that explaining why it mattered tended to make people's eyes go soft and distant, the way they did when someone at a dinner party started talking about their homebrew setup or their opinions on sourdough starters. Mervin had seen this look many times. He had seen it on the faces of venture capitalists, former colleagues, strangers at coffee shops who made the mistake of asking what he did for a living, and — with increasing frequency over the past five years — on the face of his wife, Linda, who loved him in the specific, durable way that a person loved someone whose passions they did not share but had decided, in a binding act of marital generosity, to accommodate.

"I'm not building a monument," he said to Linda on a Wednesday evening in February, standing in the doorway of what had once been the first-floor unit of their Dorchester triple-decker and was now his office, workshop, and — Linda's word — "lair." He said it preemptively, because Linda had come downstairs with a plate of leftover pasta and the expression she wore when she was about to say something supportive and devastating at the same time.

"I didn't say monument," Linda said. She was a tall woman with short gray hair and the patient, unyielding demeanor of someone who had spent twenty-three years as a principal in the Boston Public Schools and had therefore survived encounters with angry parents, budget cuts, roof leaks, a raccoon in the gymnasium, and the 2020 pandemic, all of which she discussed with

the same calibrated calm. She set the pasta on his desk, which was actually a door laid across two filing cabinets, a configuration that Mervin defended as "ergonomic" and that Linda called "the furniture equivalent of a cry for help."

"You were thinking it."

"I was thinking that you've been down here for six hours and you haven't eaten, and that the children at my school eat lunch, and they're twelve."

"I ate an apple."

"When?"

Mervin considered this. "What day is it?"

"Wednesday."

"Then the apple may have been Monday."

Linda looked at him with the particular expression of a woman who had made a cost-benefit analysis of her husband's eccentricities twenty-six years ago and had decided to invest anyway, and who now regarded her returns with a mixture of affection and exasperation, like a gardener who had planted what she'd been told was a tomato and instead grown something large, unruly, and insistent on climbing over the fence.

"Eat the pasta, Mervin."

"I will. I'm at a stopping point anyway."

This was not true. Mervin had not been at a stopping point in approximately five years. But he had learned that "I'm at a stopping point" was one of those marital phrases, like "that's interesting" or "we should talk about it sometime," that served a diplomatic rather than a literal function.

He ate the pasta — rigatoni with red sauce, Linda's specialty, good enough that Mervin periodically suggested she quit education and open a restaurant, to which she always replied that dealing with the public was dealing with the public regardless of the context, and she'd rather deal with twelve-year-olds than Yelp reviewers. He ate it standing up, looking at his screens, because sitting down felt like a concession to the idea that he should be doing something other than what he was doing, and Mervin was not, at this particular moment in his life, in a conceding mood.

What he was doing was building Basalt.

Basalt was, depending on who you asked, either the most important piece of software being written in 2026 or the most elaborate act of technological nostalgia since someone rebuilt a working PDP-11 in their basement and posted it on YouTube. Mervin had heard both descriptions. He preferred a third: Basalt was a *comprehensible system*. That was it. That was the whole pitch. A system that a single human being could understand, end to end, from the bottom of the stack to the top, with no black boxes, no inscrutable dependencies, no layers of abstraction so deep that the original intention was buried like a Roman road under centuries of accumulated city.

The name came from the rock. Basalt was the most common rock on the surface of the Earth — the bedrock of the ocean floor, the foundation of continents, formed by the simplest geological process there was: magma cooling. It was not glamorous. It was not rare. But it was everywhere, and it lasted, and if you wanted to understand the geology of the planet, you started with basalt.

Mervin's Basalt was an operating environment — not quite an operating system, not quite a framework, but something in between: a minimal, documented, fully-specified platform on which you could build software that you understood. Every component was documented. Not just API-documented — actually, narratively documented, with explanations of *why* each design decision had been made, what alternatives had been considered and rejected, and what the limitations were. Mervin had written over four hundred pages of documentation and roughly eight thousand lines of code, a ratio that was, by modern standards, clinically insane.

"The point," he had explained to Clara at a diner in Davis Square six months ago, over eggs and toast, "is that the documentation is not a supplement to the code. The documentation *is* the system. The code is just the documentation in a form the machine can execute."

"That's a very Mervin way of putting it."

"I mean it. I worked at Alphabet for eleven years. Eleven years. When I left, the codebase I'd been working on — a distributed storage system, the kind of thing that keeps your photos and emails alive across data centers on three continents — that codebase was 47 million lines. Forty-seven million.

I understood, maybe, two million of those lines — the parts I'd written, the parts I'd reviewed, the parts that the engineers I trusted had explained to me over lunch in the cafeteria that Google called a 'micro-kitchen' and that was approximately the size of a regular person's living room. Two million lines. The other forty-five million were a mystery. Not a hostile mystery — a structural one. Nobody understood all of it. Nobody could. The system was too large for any single mind."

He leaned forward. His eggs were getting cold, but Mervin's relationship with food during technical conversations was purely theoretical — the plate existed as a prop, not a purpose.

"And that was *before* AI. That was humans writing code that other humans couldn't understand because of scale. Now add AI to the equation. The AI generates code faster than humans ever could, with no inherent need to be understood, and the humans who used to understand the parts they wrote are being replaced by humans who understand the prompts they wrote, which is a different thing entirely. The code is not just too large to understand. The code is too *alien* to understand. It was produced by a process that has no concept of understanding and no incentive to be understood."

"Right now, if you look at any production codebase in any major company — your codebase, any codebase — you'll find thousands, maybe millions of lines of code that no one understands. Not 'no one has time to understand.' No one *can* understand, because the code was generated by a tool that doesn't understand it either, and the documentation, if it exists, was also generated by a tool, and the tests, if they exist, test observable behavior without verifying intent. We've built a civilization on software that works the way a Ouija board works — everyone's touching it, things are happening, but no one is steering."

"That's a little dramatic."

"Dramatic would be when one of those systems fails and nobody can fix it because nobody can read it. Dramatic would be the software dark ages — a generation from now, all the code written in 2025 is unmaintainable because the AI tools that wrote it have been deprecated and the humans who supervised it never learned how it worked."

Clara had looked at him over her coffee and said, "You know you're basically building a monastery, right? While the rest of us watch Rome fall, you're in

Dorchester with your manuscripts and your illuminations, preserving knowledge for a future that might not come."

Mervin had opened his mouth to protest and then closed it, because the metaphor was better than anything he would have come up with, and also because it wasn't wrong.

The triple-decker was on Ashmont Street, a ten-minute walk from the Red Line station, in a part of Dorchester that had been working-class Irish when Mervin and Linda bought it in 2009 and was now working-class everything, which Mervin preferred, because a neighborhood that contained a Vietnamese bakery, a Cape Verdean restaurant, a Haitian church, and a barbershop that had been in the same family for forty years was a neighborhood that had figured out something about coexistence that the rest of the world had not.

They lived on the second and third floors. The first floor had been a rental unit until five years ago, when Mervin left his job at Alphabet and Linda, in a gesture of either supreme faith or supreme resignation, said he could use it as his workspace. The unit had two bedrooms, a kitchen he'd converted into a hardware bench, and a living room that now contained three desks arranged in a U-shape, seven monitors of varying vintages, a whiteboard covered in architectural diagrams that looked like the fever dreams of a particularly organized conspiracy theorist, and a cat named Dijkstra who slept on the server rack and shed on everything.

Mervin had left Alphabet in 2021 — before the AI tsunami, before the layoffs, before the word "orchestrating" entered the vocabulary of people who had previously been content to say "managing." He'd left because he'd seen what was coming, or thought he had, and because the thing that was coming offended him in a way that he recognized was partly rational and partly aesthetic and partly something deeper that he didn't have a word for.

The thing that was coming was this: software was becoming a commodity. Not in the economic sense — it had been a commodity in the economic sense for decades. In the epistemological sense. Software was becoming a thing that existed without being known. Like electricity in a wire — you could use it, you could depend on it, but you couldn't look at it and understand it. You couldn't trace the path from intention to execution. You couldn't hold it in

your mind the way Mervin had held systems in his mind for thirty years, the way his favorite professor at MIT had taught him to hold systems — as living, logical structures that you could reason about, predict, and ultimately trust because you understood them.

Mervin did not trust things he didn't understand. This was, Linda would point out, both his greatest strength and the reason they'd argued for three weeks about the new dishwasher, which had a Wi-Fi connection that Mervin viewed with the suspicion normally reserved for a stranger lurking in one's kitchen.

"It's a dishwasher," Linda had said.

"It's a dishwasher that talks to a server in China."

"It talks to an app on my phone."

"Which talks to a server in China."

"Mervin, you built software for one of the largest corporations on Earth. You helped design systems that processed billions of data points daily. You are arguing with me about a dishwasher."

"I'm arguing with you about a dishwasher that I can't control, can't inspect, can't modify, and that will stop working the moment the company that made it decides to stop running the server. I'm arguing about a dishwasher that is not actually a dishwasher — it's a subscription to the concept of washing dishes."

Linda had bought the dishwasher. It was an excellent dishwasher. Mervin used it every day and resented it every day, and this duality was, he thought, a small-scale model of the human relationship with modern technology in general.

On Wednesday night, after Linda went upstairs and Dijkstra resettled on the server rack with the boneless fluidity that cats brought to their primary occupation of claiming surfaces, Mervin turned back to his screens.

He was working on Basalt's networking layer — the part of the system that handled communication between processes, between machines, between the inside of the computer and the outside world. It was, by any modern standard, small. His entire networking stack was about four thousand lines of

code, which was roughly the size of a single configuration file in a typical cloud-native application. It handled TCP and UDP connections, basic HTTP, and a simple message-passing protocol that Mervin had designed himself and documented in a forty-page specification that included worked examples, failure modes, and a section titled "Why Not Just Use gRPC?" which was seven paragraphs long and contained the phrase "because I want to be able to read my own code in ten years" three times in increasing degrees of italicization.

He liked working at night. The neighborhood was quiet — or quiet by Dorchester standards, which meant the occasional car alarm, the distant thump of bass from a passing vehicle, and, on memorable evenings, the operatic domestic disputes of the couple two houses down, whose arguments followed a reliable narrative arc from accusation through denial to tearful reconciliation, like a telenovela broadcast at a frequency audible to the entire block.

Tonight was calm. Mervin typed. Dijkstra purred. The code took shape on his screen — not the code that an AI would write, which was efficient and anonymous, like a sentence produced by a committee, but code that was his, that bore the marks of his thinking, that a person who knew him might recognize the way you'd recognize a friend's handwriting. He used long variable names because he believed clarity was more important than brevity. He added comments that explained not just what the code did but why he'd chosen to do it that way. He wrote functions that were short enough to fit on a single screen, because he believed that if you couldn't see a function all at once, you couldn't understand it all at once, and understanding was the point.

Was he slow? God, yes. He was spectacularly, defiantly, almost comically slow by modern standards. An AI could generate his entire networking stack in about forty seconds. It would produce code that was probably more efficient, probably more robust, probably better in every measurable way — except that no human would be able to read it and say, "I understand why this choice was made." The AI's code would be a black box that happened to be shaped like a networking stack, and Mervin's code would be a networking stack that happened to be readable, and the difference between those two things was the difference between a house and a photograph of a house.

He was aware that most people didn't care about this difference. Most people lived in the photograph and it was fine. The photograph kept the rain out.

The photograph was warm. You could raise a family in the photograph and never notice that the walls weren't real.

But eventually — and this was the thing Mervin believed with the certainty of a man who had spent thirty years watching software systems age and decay and collapse under their own incomprehensibility — eventually someone would lean against a wall, and the wall wouldn't be there.

At eleven o'clock, he checked his email. He had seventeen new messages, twelve of which were from the Basalt mailing list, which had 847 subscribers, a number that was either impressively large or heartbreakingly small depending on whether you compared it to the mailing list of a typical open-source project (heartbreakingly small) or to the number of people who cared about software comprehensibility enough to subscribe to a mailing list about it (impressively large, possibly the entire population).

One of the emails was from Clara.

Subject: Fortran Update

Mervin,

The cactus is not improving. I've moved it closer to the window but I think it might be past the point of intervention. I'm trying not to read this as a metaphor.

David announced Project Lighthouse today. AI-first engineering. "Rightsizing" the team. You called this three years ago. I owe you a beer.

How's Basalt?

— C

Mervin smiled. He and Clara had been friends since 2016, when they'd both been at a conference in Portland and had ended up at the same bar after a talk about microservices that had been so boring it had actually brought them together in shared suffering, the way prisoners of war sometimes formed lasting bonds. Clara was one of the few people who understood what he was doing with Basalt without him having to explain it, which was valuable because explaining it, as noted, tended to make people look like they'd been gently chloroformed.

He typed a reply.

Clara,

I'm sorry about Fortran. For what it's worth, cacti are more resilient than they look. Sometimes they yellow before they strengthen. (I don't actually know if this is true. I made it up to be comforting. Don't tell Linda — she already thinks I have an unhealthy relationship with metaphors.)

Basalt is good. Networking layer is almost done. Four thousand lines that I can read, explain, and defend to any human who asks. This morning I spent two hours on a TCP handshake implementation that an AI could have generated in the time it took me to open the file, and at the end of those two hours I understood my handshake the way I understand my own name, which is to say: completely, and in a way that no one can take from me.

I know this sounds grandiose. Linda says I sound grandiose approximately forty percent of the time, which I think is low but which I've learned not to dispute.

The Lighthouse thing doesn't surprise me. It will surprise you how fast it happens, though. That's the thing nobody tells you — the transition from "AI is a tool" to "AI is the worker and you are the tool" happens in the space of one executive presentation and a Sunday-evening all-hands email. Ask me how I know.

Come to the house this weekend. Linda's making her red sauce. Bring the kid if he wants. Dijkstra misses human contact with people who aren't me.

— M

He sent the email and turned back to his code. Outside, Dorchester was settling into its nighttime rhythms — the distant hiss of tires on wet asphalt, the bark of a dog, the low rumble of the Red Line pulling into Ashmont. Mervin's screens glowed in the dark room, casting blue light across the whiteboard, the server rack, the sleeping cat.

He was fifty-three years old. He had been writing code since he was fourteen, which meant he'd been programming for longer than some of his former colleagues had been alive. He'd written code at MIT, at three startups (two failed, one acquired), at Alphabet for eleven years, and now here, in a converted apartment in Dorchester, where the floor was linoleum and the ceiling leaked when it rained hard and the cat had a habit of stepping on the

keyboard and inserting random characters into his commit messages, which meant that the Basalt git log contained, among its careful descriptions of design decisions, the occasional cryptic contribution of "qqqqqqqqq33333" or "nnnnnn," which Mervin left in because they were, in their way, as much a part of the system's history as anything he'd written on purpose.

He was building something that might not matter. He knew this. He knew that Basalt might end up as a footnote, a curiosity, the software equivalent of a handmade watch in a world of smartwatches — beautiful, precise, and irrelevant to everyone except the handful of people who still believed that understanding what time it was should involve more than glancing at a screen.

But he was building it anyway. Because the alternative — accepting that software was something that happened to you rather than something you made — was not something Mervin Webb was prepared to do. Not yet. Maybe not ever.

He typed. Dijkstra purred. The code grew, line by line, comprehensible and slow and entirely, stubbornly, his.