ATROPHY

FROM TERMINALS

GILLES LOUPPE*

ATROPHY

ACT I — THE SURFACE

CHAPTER 1

The elevator in Hall C smells like someone microwaved a gym sock inside a cardboard box, which I know is two similes fighting each other in a space too small for either of them, but I don't care because I'm eighteen and I haven't yet learned that trying to describe things precisely is not the same as describing them well, but I'm getting ahead of myself, which is another thing I do.

Fourth floor. Room 412. I'm carrying everything I own in two suitcases and a backpack that's doing something to my left shoulder I'll regret at thirty, and I'm thinking: this is it. This is the part where it starts. I've been thinking in chapters lately, as though my life has an architecture I can see from above if I just squint, which is the kind of self-conscious meta-awareness that makes people want to push you down stairs, and I know this, and I do it anyway, because I'm eighteen and I think being smart is a personality trait rather than a practice, which is a mistake so common among freshmen that it should come printed on the orientation packet.

The door to 412 is propped open. I stop in the hallway. My heart is doing something fast and stupid and I stand there for maybe five seconds pretending to check my phone while actually just being eighteen in a building I've never been in, about to meet someone I'll live with for a year, and all the narrating-from-above is doing what it always does, which is keeping me from noticing that I'm terrified.

There's already someone inside.

He's sitting on the left bed — I get the right bed, I get the window, which I'm taking as an omen even though I don't believe in omens — dark curly hair doing its own thing, a flannel shirt with the sleeves rolled to the elbows, and on his desk: actual physical books. I can read the spines from here. *Structure and Interpretation of Computer Programs*. Something thick by Knuth that I'll later learn is *The Art of Computer Programming*, four thousand pages of mathematical devotion to sorting and searching. A beat-up copy of *Gödel, Escher, Bach* with Post-it notes sticking out of it like the plumage of some exotic bird of scholarship.

"Hey," I say.

He looks up. Blinks. Then he smiles. "Oh, hey. I'm Marco."

"Cool room," I say. The room is a concrete rectangle with two beds, two desks, and a window that faces another building's ventilation system. I've always been good at performances.

Marco closes his laptop. "You're CS too?"

"Yeah."

"Cool." He stands up and shakes my hand, which is unexpectedly formal and I kind of love it, the way you love anything that surprises you by being slightly out of step with its context. His grip is solid. He has calluses on his fingertips. Later I'll learn he plays classical guitar, badly, with earnest devotion, in the stairwell at 11pm until someone yells at him, at which point he moves to a different stairwell and continues, because Marco does not stop doing things because they're difficult or because someone objects; he stops doing things when he's finished with them, but right now he's just a guy in a flannel shirt who reads Knuth for fun, and I think: okay. This might work.

He'd already rearranged the furniture. Both desks pushed together to make one long surface, which meant my chair was now facing the wall. He noticed me noticing.

"Oh, sorry. I can move it back. I just, I need space to spread out. Bad habit."

He said "bad habit" the way people say it when they don't actually think it's bad. I'd learn this about Marco: he apologized for things constantly and changed nothing. It was infuriating in the way that only genuinely good people are infuriating, because you can't stay mad at someone who's sincerely sorry and sincerely going to do it again.

I drop my bags on the right bed and go outside.

. . .

The campus is concrete and glass and right angles that feel like someone tried to build a city using only graph paper and a grudge against curves. The engineering building looks like a Soviet ministry that applied for a research grant and got one. But it's September, and the light does this thing in September, comes in low and golden through the oaks along the main walk, and everything — the concrete, the glass, the students moving in clusters and alone — looks like a movie about college. Which is what I want it to look like. I want the establishing shot. The montage. The part where the soundtrack swells and the main character walks toward a future that will change everything, and I know this is narcissistic, I know that narrating your own life in cinematic terms is a species of delusion, but the delusion feels so good in September, when everything is beginning, when you haven't yet failed at anything because you haven't yet tried anything. I can hear myself narrating this, which should be a warning sign but instead feels like a feature.

I'm insufferable. I know this. I'm eighteen and I think being smart is the most important thing a person can be and I haven't yet learned that this belief is a kind of stupidity all its own, a stupidity with excellent vocabulary and no foundations. But right now, with the light through the trees and CSCI 201 on my schedule, I genuinely believe I'm about to become something. Not someone. Something. A thing that thinks. A mind that matters.

. . .

Professor Chen is small and precise. She moves like someone who's thought carefully about where to put each foot, which is to say she moves the way she thinks: deliberately, with an economy that makes every gesture feel necessary. Her handwriting on the whiteboard is immaculate — these tiny perfect characters she deploys left to right, top to bottom, no wasted space, each line a foundation for the next. Watching her fill a board is like watching someone build a house one brick at a time in a way that makes you understand what bricks are for.

"This course," she says, on the first day, standing in front of two hundred freshmen who are variously terrified, bored, hungover, or performing casualness as a defense against the terror of being somewhere that might actually matter, "is not about learning to code."

She lets that sit. Her eyes move across the room, not scanning but measuring, the way a doctor reads a waiting room. She lingers on the back rows, where the laptops are already open.

"Coding is a side effect. This course is about learning to think precisely. To take a vague problem and make it specific. To take a specific problem and break it into pieces. To take those pieces and describe them in language so exact that even a machine can follow the instructions."

She turns to the whiteboard and writes, in those immaculate letters: ABSTRACTION.

"The machine is the easy part. The machine does what you tell it. The hard part is knowing what to tell it. The hard part is the thinking."

I write this down in my notebook, a Moleskine because I'm pretending to be the kind of person who uses Moleskines, the kind of person who has thoughts worth recording in a notebook that costs fourteen dollars. I write: *The hard part is the thinking.* I underline it twice. I do not think about what it means.

Marco, who's sitting next to me, doesn't write anything. He's just watching. His pen on the desk, untouched. He's not performing the way I'm performing. He's actually listening. The difference is subtle and I can't

quite articulate it but it has something to do with the relationship between receiving and recording, and Marco seems to have a capacity for the former that I compensate for with an excess of the latter.

. . .

The first assignment goes up on Thursday. It's posted on the course page at 4pm and by 4:03 I've read it three times, each reading increasing my comprehension by exactly zero percent.

Write a Python program that takes a list of integers and returns the longest increasing subsequence.

I know what every one of those words means individually. Together they form a sentence I understand conceptually but cannot translate into action. It's like being told to paint a sunset: I know what colors are, I know what the sun is, but the brush in my hand is connected to a brain that has never converted understanding into doing, because until now understanding and doing have been the same thing — you read the textbook, you answer the question, the answer is in the textbook. This is different. The answer has to be built.

I open VS Code. I stare at the blinking cursor. The cursor blinks at approximately 500 milliseconds, which I know because I looked it up during a procrastination spiral last year that began with "how do cursors work" and ended three hours later with me reading about the history of the cathode ray tube. Right now each blink feels like a small judgment. *Blink*. You don't know what you're doing. *Blink*. Everyone else probably already started.

I type: `def longest_increasing_subsequence(nums):` and then I sit there for ten minutes while the cursor blinks at the end of the colon, and the function is named but empty, which feels like a metaphor I should use later in this very narrative but am instead noting prematurely because I can't help myself.

I go to the lecture notes. Write a for loop. It's wrong. Not wrong in an interesting way — wrong in a way that suggests I don't understand what I wrote. Delete. Start over. Still wrong.

Marco comes back from dinner and finds me cross-legged on my bed with my laptop generating sufficient heat to be technically classified as a space heater.

"The subsequence one?"

"Yeah."

"Did you try thinking about it with a smaller example first? Like, just five numbers on paper?"

I did not try this. The thought of approaching a coding problem with a pencil and paper did not occur to me, because I've been operating under the implicit assumption that coding problems are solved in code, by staring at code, by willing code into correctness through sheer ocular intensity, which is not a methodology taught in any textbook because it is not a methodology. It is desperation wearing the costume of effort.

Marco sits on his bed and pulls out an actual notebook and starts drawing diagrams. Arrows connecting numbers. Little tables with rows and columns. He's working it out by hand, slowly, and I can see him getting it wrong, backing up, crossing out, trying a different approach, and his face — his face is doing a thing I don't entirely understand. He looks happy. Not happy like entertainment. Happy like engagement. Like the struggle is the thing he came here for, the resistance against which his thinking develops force, the way a muscle needs something to push against in order to grow, and I'm watching it happen in real time, watching someone enjoy the precise activity that's making me want to throw my laptop out the window.

I try his way. I write out the numbers 3, 1, 8, 2, 5 on a piece of paper. I trace the subsequences. I start to see it. There's a pattern — not in the numbers but in how you look at them, the way you build up from each position — and suddenly the code in my head stops being random keywords and becomes a description of a process I figured out with a pencil and five numbers on a scrap of paper. Dynamic programming, except I don't know it's called that yet.

I write the code. It runs. It's wrong. But it's wrong in a new way — a way I can diagnose, because I understand what it's supposed to do, and the gap between what it does and what it should do is visible and specific, like a crack in a wall I built myself that I know how to fix because I know how the wall was built. I find the bug. An off-by-one error in the inner loop, the classic, the ancient enemy, the error so common it has a Wikipedia page. I fix it. It runs. It works.

It is 1:47 AM. I have spent six hours on a program that's twenty-three lines long. Twenty-three lines! I could write them on an index card! But I feel like I climbed a mountain, an actual vertical thing with a summit, and the air up here is thin and clear, and the understanding lives in my body, in my fingers, in the neural pathways that formed while I was struggling and crossing out and starting over, and this understanding is different from reading, different from being told — it is the thing itself, the comprehension that comes from building. I turn to tell Marco but he's asleep, his laptop closed, his notebook still open on his desk with those little arrow diagrams all over it like the notation of a dance only he knows the steps to.

I should go to sleep. I should submit the assignment and close the laptop and go to sleep.

Instead I google "longest increasing subsequence solution" because I want to see how other people did it, because the builder wants to compare blueprints, because I'm still riding the high. The first result is a Stack Overflow thread and the second result is a GeeksforGeeks page and the third result is something different.

It's a link to a conversation. Someone pasted the assignment prompt into ChatGPT and the response is there. The full solution. Clean, commented, correct. With an explanation that's better than the textbook. With a time complexity analysis I wouldn't learn about for another three weeks.

I compare it to mine. Mine is uglier. Unnecessary variables and a redundant inner condition and comments like `# check if bigger` which is the kind of comment that restates the code in English without adding information. The AI's version is elegant, professional, the code of someone

who understood the problem completely before they started typing, who had no wrong turns, no off-by-one errors, no six hours of fumbling in the dark with a pencil.

It took me six hours to write twenty-three lines that mostly work. This thing produced forty lines of perfect code in four seconds.

I close the tab. Something is happening in my chest that I can't name. It's not jealousy — you can't be jealous of a tool. It's vertigo. I've been climbing stairs in the dark and someone turned on the light and there's an elevator right next to the staircase. Same floor. And my legs are tired and the stairs suddenly look like a voluntary suffering the world has quietly made optional.

I open a new tab. I make an account. I paste the next problem from the practice set. The one I was going to try tomorrow, the hard way, with pencil and paper.

The response comes back in seconds. Clean. Correct. It explains not just the what but the why, and the explanation is patient and never gives me that look Chen gives, the appraising look that measures the distance between where you are and where you should be.

I do three more problems. Then four more. I'm not submitting these. I'm just exploring. Seeing what it can do. It's like the first time I used Google as a kid, that same vertigo of capability, that same feeling of: this changes everything. This changes what "effort" means.

My room is dark except for the laptop screen. Marco is breathing steadily in the other bed. Outside, the campus is quiet in that 3am way where you can hear the HVAC system cycling and nothing else.

I feel like I found something. Not a shortcut — shortcuts imply a destination you're still walking toward. An escalator next to the staircase, except it's infinitely tall and infinitely patient and it never breaks down. I did the hard thing tonight — I solved the problem myself — and now I know there's a tool that makes the hard thing optional. Not cheating. Cheating is copying. This is just working smarter. Everyone says work smarter, not harder. That's

literally the whole point of computer science: automate the tedious parts so you can focus on what matters. Why would I not use every tool available? And I know this reasoning has a flaw in it somewhere, I can feel the flaw the way you feel a draft in a room without seeing the open window, but the reasoning feels so good and the tool works so well and it's 3am and the capacity for self-deception at eighteen is a resource that renews faster than you can spend it.

Tomorrow I'll do the next assignment myself. Obviously. I'll use the tool to check my work, maybe. To verify. That's all. That is absolutely all.

I fall asleep quickly, the way you do when you're eighteen and you know — with the full, untested, magnificent confidence of someone who has never been wrong about anything important — that everything is going to be fine.

* * *

*
**

CHAPTER 2

The second assignment I did myself. Mostly.

Chen had introduced recursion the week before. The call tree on the board, splitting and splitting. I raised my hand, which I almost never did. "Isn't this basically induction? Base case, inductive step?" Chen's marker stopped mid-stroke. She looked at me, then wrote something in her notebook. Not on the board. In her notebook. "That's exactly right," she said. "Most people don't see that until the theory course."

I started on paper, the way Marco showed me. Merge sort this time. Split the list. Sort each half. Merge them back. Recursive. I could feel the logic propagating through the layers, each split producing a simpler problem until you hit a list of one, the base case, the bottom turtle.

I wrote the code and it was wrong and I fixed it and it was wrong differently and I fixed that and it sorted and I felt the mountain thing again, smaller this time, a foothill, but still mine.

Except the merge step was wrong. Two sorted halves going in, garbage coming out. I stared at it for forty minutes, swapping indices, reversing comparisons, printing intermediate states that made no sense. The merge worked on paper. In code it didn't. I could see both halves were sorted. I could see the merge was comparing the right elements. But the result was scrambled, and the gap between what I understood and what the code

did was maddening because I understood the algorithm, I really did, I just couldn't translate the understanding into working instructions, which is exactly the gap Chen described on day one, the gap between knowing and telling, and I was standing in it at 11:17 with the assignment due at midnight.

I opened ChatGPT. Just for this one thing.

"My merge sort merge step produces wrong output even though both halves are sorted correctly."

I pasted my code. The response came back in seconds. I was mutating the original list while reading from it. The merge was overwriting elements it hadn't compared yet. I needed a copy. One line: `left, right = nums[:mid], nums[mid:]`. Then merge into the original. The issue wasn't the algorithm. It was the data. I was destroying my own input.

I knew about aliasing. I knew about mutation. I just couldn't see it in my own code, because seeing it requires holding the whole execution in your head at once — what the code says, what the memory looks like, how they diverge — and that's a skill, not a fact, and I hadn't built it yet. The tool built around it.

I pasted the fix. Not the whole solution. Just the concept, rewritten in my own syntax, with my own variable names, which I told myself mattered, which I told myself constituted authorship, in the same way that translating a sentence from French to English constitutes authorship — you chose the words, even if someone else composed the thought. I submitted at 11:41. Full marks.

The guilt was a specific shade of gray. Not transgression. Compromise. The kind that doesn't announce itself but changes the color of everything slightly. The line was so thin, and the other side looked so much like this side.

I told Marco about it the next morning, framing it casually — deploying the confession as evidence of its own insignificance.

"I used ChatGPT to figure out an edge case last night."

He looked up from his cereal. "For the merge sort?"

"Yeah, just a mutation thing. I'd been stuck for like an hour."

He shrugged. "I hit the same thing. I ended up drawing out the memory layout on paper until I could see where the overwrite happened. Took me like two hours."

Two hours. He spent two hours on something I solved in thirty seconds with a prompt. His grade would be the same as mine. Tomorrow neither of us would remember the specific fix. But somewhere in Marco's brain there was a groove being worn deeper, a neural pathway that knew how recursive base cases felt, not as facts to be retrieved but as patterns to be recognized, the way a carpenter knows when a joint is tight by feel rather than measurement. The kind of knowledge that doesn't show up on a transcript. The kind that shows up the next time you see a base case, and the time after that, accumulating silently like interest in an account he doesn't know he's building, compound and invisible.

I didn't think about this. I noted it the way you note a change in weather and incorporate it into no plans. I ate my cereal. Marco was doing the thing he did where he ate with one hand and read with the other, except the book was propped against the milk carton and kept sliding. He caught it for the third time and said, without looking up, "I need a book stand."

"You need to stop reading at breakfast."

"You need to stop telling me what I need."

He was smiling. I was smiling. The kind of exchange that means nothing and means everything, the private language of people who live in the same small room and have started to sync. We walked to class. The October sun was golden and I felt fine. The guilt had faded to something indistinguishable from the general background noise of being a person with a conscience who occasionally does things that the conscience notes and then forgets.

· · ·

By November I had a system: try first, then ask.

I gave myself thirty minutes. If I couldn't crack it, I opened the chat. I was not one of those kids who pasted the assignment prompt straight in — I saw them in the library, copying output with a find-and-replace on the variable names. That was cheating. What I did was different.

At least that was the story. And the story worked because it was mostly true, the way effective lies are mostly true. The part that's false is the part that matters most. I was learning. When the AI explained a concept, I read the explanation. I followed the logic. I nodded along. I thought I understood. The understanding had a texture, and if I had been more honest I would have examined it more carefully: smooth, frictionless, like reading a Wikipedia article about skydiving. Informative. Comprehensive. Utterly different from jumping out of a plane. But who jumps out of planes when you can read about it?

The grades were good. Very good. My code had developed a professional sheen that came from absorbing the AI's style — its naming conventions, its preference for list comprehensions over explicit loops, its elegant error handling with custom exceptions and retry logic. The TAs wrote things like "excellent implementation" and "clean, well-documented code." I was, by every metric the university cared to measure, excelling. The metrics were correct. The metrics were measuring the wrong thing.

Marco got a B on the midterm. I got an A-minus. He'd spent the week before filling a notebook with practice problems, working them by hand, getting half wrong, then figuring out why they were wrong, which took longer than solving them would have, and at the end of it his understanding was as dense and load-bearing as a foundation poured by hand, lumpy and imperfect and absolutely solid. I spent the night before having ChatGPT explain the key algorithms with examples, then quizzed myself by trying to explain them back. A rehearsal technique. A performance of preparation, and the performance was convincing enough to convince me.

The difference showed in class. When Chen asked a question off-script — something lateral, requiring you to connect two ideas in real time — Marco raised his hand. And I didn't. Not because I didn't know the answer.

Because by the time I'd assembled a response from the parts I'd memorized, Marco was already three sentences into an explanation that sounded like he was building it as he spoke, connecting it to something from two lectures ago that I'd forgotten. Chen's pen stopped moving when Marco talked. She didn't write down what he said. She wrote down something else, something shorter, and I never thought about what it might have been.

What I was going to say felt thin. Rehearsed. Like quoting someone else's insight and presenting it as original thought.

I started sitting further back. From the back, Marco's voice was just another sound, and Chen's questions were just prompts I didn't have to answer. Once, early in the semester, she stopped mid-lecture and said, "I can see your screens from here." Nobody laughed. She didn't say it again.

The semester ended. Dean's List. GPA: 3.8. My mom cried on the phone, that particular kind of crying where pride and relief converge into a sound that isn't quite either, and I felt the weight of her pride like a warm hand on my shoulder and I thought: I earned this. I did the work. The AI was just a tool. Everyone uses tools. Architects use CAD software. Mathematicians use calculators. Writers use word processors. I'm just doing what everyone does, with a slightly more powerful tool, and the difference between a tool and a crutch is the difference between choosing to use it and needing to, and I choose to, the choice is entirely mine.

I went home for winter break and everyone told me how proud they were and I felt proud and the pride sat well, and I didn't think about the thirty-minute rule, which by December had become a twenty-minute rule, which by the last assignment had become something more like: try, but not too hard. Not past the point where it stops being fun. Not past the point where I could just ask.

* * *

CHAPTER 3

Spring semester. The tool got better. Or I got better at using it. Both things were true, and the distinction between them mattered, and I treated it as though it didn't.

I discovered Copilot in January. ChatGPT was across town — you had to go to it, ask, bring the answer back, and the journey imposed a friction, a gap, a moment of air between the problem and the solution during which you might, if you were paying attention, think for yourself. Copilot eliminated the gap. It was in the editor. In the room. Watching me type. Finishing my sentences, except it was always right and never annoyed. The first time it autocompleted an entire function — correctly, from just the function name and a comment — I actually laughed out loud. A startled, delighted, slightly unhinged laugh, the laugh of someone who has just witnessed a magic trick and understood the mechanism and found both the trick and the mechanism independently wonderful.

Marco looked over from his desk. I showed him. He squinted at the screen, read the autocompleted function, and said, "Huh." A single syllable. Then he went back to his textbook. *Introduction to Algorithms*, 1,300 pages. He was four chapters ahead of the class, highlighting passages and making margin notes in pencil so small they looked like the footnotes to someone else's thoughts. Those extra chapters were leaving deposits in him, layers of

understanding accumulating the way silt builds in a riverbed, each layer too thin to see but collectively forming something solid.

I didn't read the textbook anymore. I skimmed the section headers, then let the AI summarize the important parts. The summaries were concise, clear. I didn't need the proof. I needed the pattern.

February: I stopped going to Professor Chen's office hours. There was no point. Whatever question I had, the AI answered faster, without the social choreography of pretending I'd tried harder than I had, without Chen's appraising look, the look that measured the distance between the question I asked and the question I should have asked.

March: A group project. Four of us building a web scraper in Python. I wrote my part in an afternoon — which was actually me describing what my part should do in a carefully worded prompt to Claude and then refining the output through three rounds of conversation. The code was excellent. It handled edge cases I wouldn't have thought of. It had error handling with custom exception classes and retry logic and logging that made the whole thing look like the work of a senior engineer with a decade of experience. I read through it line by line and understood what it did, more or less, the way you understand an engine when someone opens the hood and points: that part spins, this part pumps, fuel goes in, motion comes out. I could explain it. I could not have written it from scratch. But nobody asks you to build an engine from scratch.

My teammate Dana was struggling with the API integration for three days. I took her description of the problem and pasted it into a new chat and gave her back the solution in fifteen minutes and she looked at me like I'd performed a magic trick. "How did you figure that out so fast?" she asked, and I said, "I've seen this pattern before." I had seen it. The AI recognized it.

Marco wasn't in my group. He was up until 3am three nights running. Their project was less polished. They had a bug in the HTML parser — self-closing tags treated as opening tags, the tree growing deeper, the stack never unwinding. When they found it, Marco described the hunt at midnight while making pasta, his eyes lit up.

"I literally printed every node as it was created and just watched the tree grow wrong." He laughed. "Two hundred lines of debug output. I taped them to the wall and walked along them like I was reading a scroll."

He talked about code the way musicians talk about difficult passages. With affection. With intimacy. Like he and the code had a private language, a shared history built on struggle and the occasional breakthrough that justified all the struggle. A relationship.

I realized I didn't talk about code at all. I talked about projects. Deliverables. Outputs. The code was a means and the end was the grade and the grade was good and so everything was good. I couldn't describe my own code from memory. Couldn't trace the flow of data through the functions. It was my name on the commit but the code could have been anyone's.

April: The thirty-minute rule was dead. I didn't remember when it died. At some point I realized the rule was arbitrary anyway. Why thirty minutes? Why not twenty? Why not five? The point was never the time. The point was understanding the solution, and I understood the solutions. I just got there faster now.

Now I read the problem, thought about it for maybe five minutes — enough to develop a sense of what the answer should look like, a silhouette — and then I prompted. Sometimes I skipped the five minutes. It always worked. That's the thing nobody warns you about. Not that it will fail. That it won't.

End of freshman year. Dean's List again. GPA: 3.85. My mom was so proud she put the letter on the fridge next to a photo of me at age six holding a Lego spaceship I'd built from the instructions, which is a kind of prophecy or a kind of warning, depending on how you feel about the distinction between following instructions and designing the thing the instructions describe.

Marco had a 3.5 and bags under his eyes and a GitHub profile full of messy, personal, heavily commented projects that read like the diary of someone learning to think. His commit messages told stories: "Finally fixed

the damn parser. The issue was self-closing tags. I am never looking at HTML again. (Lie.)" Each commit was a fingerprint, a record of a specific human being making specific decisions, getting things wrong and then right, and the wrongness and the rightness were both visible, the way scars are part of a body.

I had a GitHub profile full of clean, anonymous, professional code. Same process every time. Not mine.

Summer. Home. Slept late. Played video games. Scrolled through my phone in bed for hours, the scroll a reflex now, a tic, the thumb moving on its own, content arriving and departing like trains I watched through a window without boarding. I didn't code. Not once. Not for fun, not for practice. It didn't occur to me. The absence of an impulse is the most invisible thing in the world. You don't notice the hunger you never developed. You don't miss the itch you never felt. You don't mourn the curiosity that was supposed to grow here, in this space, during these months, because you don't know it was supposed to grow.

I spent the summer recovering. From what?

Marco texted in July. A photo of his laptop screen: he'd built a chess engine in C over the summer. Pure C. "It's terrible," he wrote. "It loses to everything. But the bitboard representation is mine and it works and I love it."

I texted back a thumbs up. I meant it. I genuinely, sincerely meant the thumbs up. It was an honest reaction, the authentic response of a person who was happy for his friend. What I couldn't explain, even to myself, was why I kept going back to the photo. Not the code. The expression on his face reflected in the laptop screen, barely visible in the corner. He looked the way I looked at 1:47 AM freshman year, twenty-three lines of ugly code running clean. I scrolled past it. I went back to it. I put the phone down. I just had nothing equivalent to offer in return. No project. No creation. No thing I'd built because I wanted to, because the building was the point, because the C was hard and hard was the reason. My summer was a blank. A pleasant, comfortable, well-rested blank.

I started typing something longer. Something about how I'd been thinking about trying Rust, or maybe building a small game, and I got four words in and deleted them because they weren't true and Marco would know they weren't true. He had that quality. You couldn't perform for him. He'd just look at you with those patient eyes and wait for the real version.

The thumbs up sat in the chat. Marco sent a smiley face. We moved on.

. . .

ACT II — THE SHORTCUT

⁂

CHAPTER 4

Sophomore year started and something had shifted. Not in me. In everyone. September again, and the light was doing its thing through the oaks, and I noticed it the way you notice a song you used to love playing in a grocery store — familiar, pleasant, someone else's.

You could feel it in the library first. The library was different. The study rooms that used to sound like frustrated sighs now sounded like calm, productive silence. Everyone had their tools open. Not hidden. The freshman whisper — *is this cheating?* — was gone. We'd decided it wasn't.

If everyone is in the elevator, the elevator is the floor. You'd be stupid not to get in.

Data Structures. Operating Systems. Algorithms II. The courses got harder. The tools got better. Claude was my primary now. It didn't just solve problems. It anticipated why I was asking. What this produced wasn't relief. It was trust. The kind that replaces thought.

I had a new workflow. Read the assignment. Paste it. Get the solution. Read the solution. Make sure I "understood" it — which meant: could I explain this at a party? Could I wave my hands at the big idea and make the right sounds? If yes, submit. If no, ask for a simpler explanation, then submit.

There was a kind of efficiency to it that felt like intelligence. I was processing information faster than anyone in my cohort. I was an optimizer, the system was being optimized, and the outputs were excellent.

The understanding was real, in its way. I knew what a B-tree was. I knew why hash tables were fast. I knew BFS from DFS. I knew these things the way I know mitochondria are the powerhouse of the cell: facts, sealed and labeled, retrievable. I could say the words. I couldn't open the package.

Marco knew differently. He built a B-tree from scratch over fall break. It took him a week. The first version was wrong in four different ways: split logic backwards, keys not propagating, leaf pointers circular, the whole thing segfaulting on inputs larger than 64 elements. He found and fixed all four bugs, one at a time, and by the end he could draw the insertion procedure on a napkin from memory. Including the split. Including the edge cases. The napkin looked like Marco's brain turned inside out. It was correct.

I could tell you what a B-tree does. Marco could build you one blindfolded.

The difference is invisible on a transcript. It's invisible on a resume. It's invisible to every metric the university uses to sort students into categories of success. The only place the difference shows is in a room with a whiteboard and a marker and no Wi-Fi. That room hadn't arrived yet.

One night I came back from a party around midnight. Marco was at his desk surrounded by crumpled notebook pages. He looked at me, then at the clock.

"Don't you have the algorithms midterm tomorrow?"

"Yeah. I studied earlier."

He looked down at his notes, the hours of work spread across his desk. Then he looked at me again and something crossed his face I hadn't seen before. Not envy. Something more like confusion. Like he was running numbers that didn't add up.

"How do you have time for everything?" he asked. Not accusing. Genuinely confused. Like he was looking at an equation that should balance but didn't.

I shrugged. "I'm efficient."

He nodded slowly. Then he did something he'd never done before: he closed his notebook while I was in the room. Like the work was suddenly private. Like my presence next to it made it look different.

"Night," he said.

"Night."

Whatever that was, I let it pass. Easier than looking at it. I went to bed.

But we weren't graded on building in the dark. We were graded on output, measured in correctness and rubric compliance. By those measures I was winning. I got an A in Algorithms II. The professor wrote: "Sophisticated implementation. Clear understanding of trade-offs." I read that and felt warm.

I went to a lot of parties. I had time. The hours I used to spend struggling were free, and I filled them with beer and Netflix and the kind of socializing where everyone's in the same room looking at different phones. It was fun. It was what college was supposed to be.

The free time was strange. I didn't do anything with it. Didn't code for fun. Didn't read for fun. Didn't start side projects. The time was free and it stayed free. Like a room with no furniture you stop entering.

* * *

CHAPTER 5

Junior year. I stopped going to lectures.

It happened in stages. First the 8ams. Then Mondays. Then I realized I could get ninety minutes of lecture in ten minutes of conversation with Claude.

I was optimizing. That's what smart people do. Find the bottleneck. Eliminate it.

I still watched some recorded lectures at 2x speed while eating. The professors' voices came out fast and compressed, content stripped from its packaging.

Marco still went to every lecture. Third row. Handwritten notes. He asked questions that made professors pause. I knew this because Dana mentioned it. "Marco asked this wild question in OS today and Chen just stood there for like thirty seconds before answering." I nodded. I wasn't there. I didn't ask what the question was.

The thing about lectures is they're inefficient. Ninety minutes for material you can get in ten. A professor going on tangents about some paper from 1978, students asking questions you already know the answer to. You sit there waiting for the one sentence that matters. I was cutting the fat. Getting the signal without the noise.

I thought about what I was gaining: time.

I filled it with nothing.

My world contracted. Not geographically. But the intellectual world, the part made of curiosity and not-knowing, was getting smaller. I used to read random arXiv preprints. I'd see a title like "Attention Is All You Need" and fight through the notation and understand maybe 40%. But that 40% would rearrange something. It was exercise.

I didn't do that anymore. If I wanted to know what a paper said, I pasted the PDF. The summary came back in thirty seconds. Easier than thinking.

That semester the tools changed again. Not better the way Copilot was better than ChatGPT. Different. Someone showed me an agent framework in a Discord server. You described what you wanted — not the code, the goal — and the agent planned the steps, wrote the code, ran it, hit an error, read the error, fixed the code, ran it again. The whole loop. The struggle loop. The one I used to do freshman year with a pencil and five numbers on a piece of paper.

I watched it work through a problem set. It tried an approach, hit a wall, backed up, tried another. It debugged. It iterated. It did the thing Marco did, the thing I'd stopped doing, except it did it in seconds, silently, without the stairwell guitar or the crumpled notebook pages or the midnight pasta. Without the joy.

I didn't think about the implications. I just used it. The last friction disappeared. I didn't even prompt anymore, not really. I described. I said what I wanted and the machine figured out how to build it, including the parts I wouldn't have known to ask for. The distance between intention and artifact collapsed to zero. I was the client. It was everything else.

There was a moment in October. The library, one of those long tables by the windows. Two freshmen at the next table. One was stuck — her whole body curved into the screen. The other was helping, drawing diagrams on scrap paper.

The stuck one said, “Oh, wait.” She stopped. Her face changed. Eyes widened. She looked at the scrap paper, then her screen, then the paper again. “Oh wait, I see it. The invariant holds because…”

Her whole face changed. Something opened up in it. I watched her look at the paper, then the screen, then the paper again, and she was somewhere I recognized but couldn’t get back to.

I tried to remember the last time my face did that. Not nodding at an explanation. The actual thing.

It was freshman year.

After that, the moments stopped. Not suddenly. I just stopped noticing when the last time was.

I noted this. I filed it. I opened my laptop. The Moleskine from freshman year was on my desk, under a stack of printouts. I moved it to make room. There was an assignment due.

* * *

*
**

CHAPTER 6

The pattern was stable. Assignment in, prompt out, review, submit. Thirty minutes start to finish.

Five classes. A's in four. A research position in the ML lab, the kind of thing that looks great on a resume. It mostly meant running experiments Claude helped me design and writing reports Claude helped me draft. Dr. Patel published six papers a year and needed hands to run the pipeline. My hands were fast. My outputs were clean. He told me I had "great instincts for experimental design." I glowed for twenty minutes.

My resume filled up. On paper I was the student I'd set out to be.

Marco got a C-plus on the OS midterm. He was frustrated for a day. Then he went to Chen's office hours and sat with her for ninety minutes going through every wrong answer. Not just what the right answer was but why he got it wrong.

I passed her office on the way to the vending machine. The door was open. She was leaning forward, drawing something on a piece of paper, and Marco was frowning at it, and it looked like an argument between two people who respected each other enough to disagree carefully. Her office hours sign said 3:00-5:00. It was almost 7.

He came back and said, "I was thinking about virtual memory completely wrong. Fundamentally wrong. But now I see it." He sounded grateful for the C-plus.

I got a B-plus on the same exam. I pasted my wrong answers into Claude. It explained them. I read the explanations. Nodded. Moved on. The difference between how I processed my wrong answers and how Marco processed his was the difference between driving past a town and living in it.

One weekend that semester, Marco came back from the library looking raw. Not tired. Something else. He dropped his backpack on the floor and sat on the bed staring at nothing.

"You alright?"

"Yeah. Just." He rubbed his face. "I've been doing the OS practice problems for six hours and I still can't get the page replacement stuff. Everyone else just asks Claude and they're done in twenty minutes and I'm sitting there like an idiot drawing page tables by hand."

He looked at me. Not accusing. Just tired.

"Sometimes I wonder if I'm doing it wrong. Like, what's the point of spending six hours on something when the answer's right there."

He pulled at a thread on his sleeve. He did this when he was stuck on a problem. I'd never noticed until now.

"Maybe I'm just—" He pulled at the thread again. Stared at his notebook. "I don't know."

He was quiet for a long time. Then he opened his notebook again, but he didn't write anything.

I almost said something. I could feel it forming — something about freshman year, about the merge sort, about how he'd sat on his bed drawing arrows while I stared at a cursor. Something true. But the words wouldn't organize. I'd gotten out of the habit of saying things I hadn't rehearsed.

"You'll get it," I said. I was thinking that he could have been done in twenty minutes.

He went back to the page tables. By midnight he had it. I could hear him muttering the algorithm to himself, testing it against examples, tracing through the edge cases. By morning he could do it cold. The six hours had purchased something. I couldn't have told you what.

The OS syllabus changed three times that semester. First version was standard: two midterms, a final, problem sets. Second version went up in September. Chen had added "In-Class Demonstrations." Every other week, randomly selected students would solve a problem on the whiteboard during lecture. No notes. Fifteen percent of the grade.

The class GroupMe lit up. "Is she serious?" Someone posted a Reddit thread about professors who do whiteboard cold-calls.

Third version came in October. The demonstrations reduced from 15% to 5%. A note at the bottom, in italics: *"Assessment format adjusted following departmental review."*

Dana told me what happened. "A bunch of students complained to the dean. Said the whiteboard thing was unfair. Chen pushed back but the department made her scale it down."

"She's trying to catch people using AI," Dana said. "That's what it's about, right?"

"I guess."

The 5% whiteboard component happened twice. The first time, four out of six students couldn't get past the first step. The second time, she called on a student who wrote a near-perfect solution, then asked him to explain the time complexity, and he froze. Clean code on the board. No words in his mouth.

After that, the whiteboard section disappeared from the syllabus entirely. No announcement. Just gone.

I noticed because my grade calculation changed by 5%.

Something else changed that semester, quietly. The problem sets got easier. Not obviously — the problems still looked hard. But the proofs were gone. The questions that used to say "show why this invariant holds" now said "implement this algorithm and test it against the provided cases." Output you could check. Output a machine could generate. The assignments that would have exposed the gap were gone because the gap was too wide and the complaints too loud.

The system adapted. The problem sets got easier. The grades went up. Nobody complained. After a while nobody was looking.

Spring. Software engineering. A project course. My team loved me. Fastest coder they'd worked with. Most productive member, measured in pull requests and features per sprint. When someone asked why I chose a particular design pattern, I gave an answer that sounded right. Observer pattern for notifications. Repository pattern for the data layer. The answers were correct. They just weren't mine.

One evening I was scrolling through Chen's course evaluations. Most were what you'd expect. One stood out:

"Professor Chen clearly knows her stuff but her expectations are unrealistic. Nobody does proofs by hand anymore. The whiteboard exercises felt like punishment for using tools that every company in industry expects us to know."

Forty comments saying the same thing in different words. One didn't:

"Best professor I've had. She actually makes you think. I hated every second of it and I learned more than in any other class."

That one was probably Marco.

* * *

CHAPTER 7

The semester ended the way semesters do.

Summer between junior and senior year. Internship at a mid-size tech company. Open office, standing desks, a coffee machine that cost more than my car.

Copilot in the editor, Claude in the browser, an agent running in the terminal. A ticket came in. I read it, pointed the agent at the relevant codebase and the ticket description. It explored the code, found the right files, made the changes, ran the tests, fixed what failed, ran them again. I reviewed the diff. Looked right. Pushed. Review came back with minor notes. I pointed the agent at the review comments. It addressed them. Shipped.

My manager liked me. "You ramp up fast," she said. The agent ramped up fast. I was the layer between it and the organization. A name on the PR. A face in standup. I shaped things a little: a commit message with a joke, a Slack emoji on someone's PR. Human fingerprints on machine work.

The work was easy. CRUD apps, API endpoints, bug fixes. Standard stuff. The tickets flowed in, processed, out. I hit every deadline. I was what they wanted.

One afternoon there was a production issue. The payment service was dropping transactions intermittently. My manager asked me to look into it. I pulled up the logs, read the errors, pasted them into Claude. Claude

suggested checking the connection pool settings. I checked. That was it. Connection pool was too small for the traffic spike. I fixed it. Took forty minutes, most of it waiting for deploys.

My manager said nice work. It was nice work. Clean diagnosis, fast fix.

Later, in the kitchen, two senior engineers were talking about the same issue. Why pool exhaustion manifested as dropped transactions instead of timeouts. How the retry logic interacted with the pool recovery. Whether the monitoring should have caught it earlier. They talked for twenty minutes. They weren't fixing the problem. It was already fixed. They were understanding it.

I listened. I had nothing to add. I'd fixed the issue. I couldn't explain why the issue existed.

My code was clean the way hotel rooms are clean. Anyone could have written it. No one did.

The internship ended in August. "Exceeds expectations," my review said. "Strong contributor. Fast learner." Both statements were true.

Back to school. Senior year. Marco had been at a startup over the summer. Eight engineers, no AI policy, just people writing code. He'd built a search feature from scratch. When he described it he used words like "elegant" and "tricky." He spent three days on one function, edge cases in a query parser. Four lines shorter than the previous version. Twice as fast. He was proud of it the way a blacksmith is proud of a clean weld.

I couldn't imagine spending three days on one function.

I couldn't imagine.

. . .

ACT III — THE VOID

⁂

CHAPTER 8

Senior year. September again. The light through the oaks.

I had a 3.9 GPA and six projects in my portfolio and a research position and an internship and letters of recommendation. I could describe each project in a sentence. Only a sentence. If you asked for two, the second would be the first one rephrased.

Chen taught the capstone course. Same precise movements. Same block letters. She announced the midterm would have a whiteboard component. No laptops. No phones.

"This isn't about getting the right answer," she said. "It's about showing me how you think."

I prepared with Claude. Practice problems every night for two weeks. Greedy: sort, iterate, track. DP: subproblems, table, trace back. Graphs: BFS for shortest, DFS for components. Templates in my head like lines in a play.

The day came. Small room. Fluorescent lights. A whiteboard. Chen sat by the door with a clipboard. She'd been at it for two hours. I could see check marks and notes on the page above mine. Some pages had a lot of writing. I picked up the marker.

"Design a data structure that supports insert, delete, and get-random, all in O(1)."

I knew this one. HashMap plus Array. I wrote it on the board.

"Walk me through insert."

"Add the element to the array. Store its index in the map."

"Delete?"

"Swap target with last element. Update the map. Remove last."

"Why does naive deletion take O(n)?"

"You have to shift elements."

"Right. And what invariant does the swap maintain?"

The word "invariant" landed on something hollow. I knew what an invariant was. I couldn't see the one she meant.

"The array stays packed," I said. "No gaps."

Chen nodded. Made a note. We moved on.

Two more problems. My answers were correct. My explanations were thin. You knock on the walls and the sound comes back hollow.

She didn't ask me to explain further. With some students she pushed for twenty minutes. With me she moved on quickly.

Walking home after. November air cold on my face. I thought about standing at that board. The marker in my hand. The answers coming from storage, not construction. Like pulling files from a cabinet versus building the cabinet. I knew the words. I didn't know what they meant.

. . .

I needed to print something. Third floor printer was broken. The fourth floor one was near the faculty offices.

I walked past the row of doors. Chen's was half open.

She was talking to someone. Patel, I think. I wasn't paying attention until I heard "the students."

"I can't fail them," Chen was saying. I slowed down. The printer was right there.

"The work is correct. The code compiles, the tests pass. I gave a whiteboard exam and most of them could reproduce the algorithm. Reproduce it. Like reciting a poem in a language you've memorized phonetically."

I picked up my printout.

"And their writing. First-years come in messy but alive. They write these tangled paragraphs where they're clearly wrestling with ideas too big for them, and it's — " A pause. "By senior year the reports are correct. That's the word I keep using. Correct. I hate that word. It used to mean something good. But it's not just the reports. Their emails. Their questions in class. Everything gets shorter, flatter, more… functional. They stop reaching for words."

"I do live coding in class. I pause at the cursor. They used to shout suggestions. Now they just wait. The whole room waiting for text to appear, like the cursor is a loading icon."

I walked toward the stairs.

"It's not cheating. That's what makes it hard. They're not breaking the rules. They're just not developing. The muscle never forms. I'm watching atrophy in students who've never been strong."

A pause.

"I keep thinking about this student I had freshman year. Sharp kid. Asked a question once about recursion and induction that made me stop. He's still in my class. Senior seminar. His work is flawless. I have never been less interested in a student's work."

I went down the stairs. She was talking about me. Obviously not me specifically. She had hundreds of students.

I was thinking about the printer and whether the color cartridge was low because my diagrams looked faded.

Later that night I thought about it briefly. Chen talking about students. The thing about perfect answers. It sounded like something you'd read in an op-ed. Professors always think things were better before. Every generation says the next one is worse.

I opened my laptop. There was a ticket to close before tomorrow.

* **

CHAPTER 9

Interviews started in November.

I prepped with LeetCode and Claude. Two hundred problems in three weeks. My version of "doing" a problem: read it, prompt it, read the solution, make sure I knew the pattern. Two-pointer. Sliding window. BFS. The patterns were in my head like cards in a deck.

The coding screen went fine. Medium-difficulty problem. Twenty minutes.

The on-site was in December. Four rounds in a glass building. System design went well. Vocabulary and frameworks: load balancer, caching, sharding, message queue. The words came out in the right order.

Third round. A senior engineer. Short gray hair. My portfolio on her screen.

"Walk me through this project." The web scraper from sophomore year.

"Web scraper for e-commerce sites. BeautifulSoup for parsing. Handles pagination and rate limiting."

"I can read the README," she said. "Tell me about the rate limiter. Why token bucket over leaky bucket?"

I looked at the screen. There was a class called `TokenBucketRateLimiter`. I didn't choose the name. I didn't choose the algorithm.

"Token bucket gives burst capacity," I said.

"Right. But for web scraping you're trying to mimic human browsing. Wouldn't leaky bucket's steady rate be better?"

"That's a good point."

Silence. She was giving me space. To show I'd thought about this. To demonstrate that there was something behind the answer besides the answer.

I didn't have more.

The rejection came two weeks later, a single paragraph. "We were impressed by your technical skills but are looking for candidates who demonstrate deeper architectural thinking."

I read it twice. Not because it hurt. Because I was waiting for it to hurt.

I got another offer. The interview was shorter. Two rounds. More practical questions. They wanted someone who could write clean code and ship features. I could do that.

Marco got hired by a startup. Take-home project, then a ninety-minute call where he walked them through every decision. They offered him the same week.

* * *

CHAPTER 10

Last semester. Capstone project. I built a machine learning pipeline for sentiment analysis. It worked. Clean documentation. Modular code.

The project took three weeks. Most of that was waiting for models to train. The actual coding was two days of prompting and adjusting.

Chen posted final grades on a Friday. I got an A. The comment field said: "Technically sound."

Two words. The student before me on the grade sheet, visible for a moment before the portal anonymized, had a paragraph. I scrolled past.

Graduation was a week away. I went to return a library book and took the long way through the CS building. Chen's door was open. She was packing a box. Books, papers, a framed photo from the wall, the one with laughing students around a whiteboard.

"Moving offices?" I said. I was being social. The ritual.

"Retiring," she said. She didn't look up. "End of the semester."

I didn't know what to say. I said, "Congratulations."

She looked at me then. Not the appraising look from freshman year. Something quieter.

"You know, when you started here, you asked a question in class once. About the relationship between recursion and mathematical induction. I

wrote your name in my notebook. I thought, this kid is going to be interesting."

I don't remember this.

"I've been teaching thirty-two years. The last four have been…" She picked up a paperweight, turned it, set it down. "I used to be able to tell who understood and who didn't. Now everyone's work looks the same. Polished. Correct. Empty. I can't tell the difference between a student who understands deeply and one who typed a prompt carefully, and I've spent four years trying to design an assessment that can, and I failed. So."

She put the paperweight in the box.

"I'm not angry at you. At any of you. You're rational. The tools are there. The incentives point one way. I just can't…"

She picked up a book from the box, looked at the cover, put it back.

"I used to have students who would come to office hours just to argue with me. Not about grades. About ideas. I had one student who spent three office hours trying to convince me that Turing completeness was a limitation, not a feature. She was wrong. She was so productively wrong. I still think about that argument."

She looked at the box.

"I can't stand in front of a room and teach people to think when the room has already decided thinking is optional."

I nodded. I said something about how much I'd learned in her classes. The words came out smooth. Appropriate. The kind of thing you say.

For a second I wanted to say something real. I could feel it, the shape of it, somewhere behind my sternum. Something about freshman year, about the question I asked that I don't remember asking. About how she wrote my name in her notebook and I became a person who wasn't worth writing down.

The feeling passed. They pass quickly now.

She smiled the way you smile at someone who's just proved your point.

She knew. She'd always known. And the knowing wasn't enough to keep her there.

. . .

ACT IV — THE BOTTOM

⁂

CHAPTER 11

Graduation was in May. My parents drove up. My mom took photos. The cap. The gown. The handshake. My dad said he was proud. My mom asked if I was eating enough. She always asks if I'm eating enough. It's the question underneath all her other questions. I said yes. I was eating fine.

The diploma said Bachelor of Science in Computer Science.

The job started in June. Good company. Good salary. Open office, free lunch.

Day one. Laptop. VS Code. Copilot. Claude. The setup took an hour. The tools were right there. They were always going to be right there.

First ticket. A bug in authentication. Session tokens not expiring properly.

I read the ticket. Found the code. Pasted it with the bug description. The fix came back. I read it. Made sense. Implemented it. Pushed.

"Great start," my manager wrote. Rocket emoji.

Day two. Another ticket.

Day three. Another.

The loop: ticket, read, prompt, adjust, push, review, ship.

Smooth.

. . .

Weeks passed. I was productive. I hit deadlines. Clean PRs. Good standups.

One day there was a design review. Database migration. Seven engineers in a conference room. Whiteboard at the front.

People talked. Drew diagrams. Argued about backward compatibility and rollback strategies and data integrity during the transition window. They had opinions. Strong ones. Built from experience and from having been wrong before.

My manager asked what I thought.

I said something about doing it in phases. Reasonable. She nodded. Someone else said the same thing with more detail, specific steps, knowledge of why certain approaches fail.

I sat through the rest of the meeting. I agreed with things. The things were reasonable.

My hands were under the table. I don't know when I put them there.

After the meeting I went back to my desk. Opened Claude. Asked about database migration strategies. The answer covered everything the meeting had and more. I could have said all of that if I'd had my laptop open.

But the meeting was a whiteboard. And my thinking was in the laptop.

. . .

I went home. The apartment was fine. One bedroom. Clean, impersonal, no evidence of the inhabitant. The hallway smelled like the neighbor's cooking — garlic, something fried. Quiet.

I made dinner. Or I ordered it. I ate on the couch.

I opened my phone.

Scrolled. Social media. News. Short videos. A thumb movement. Content arrived and I reacted and more arrived. Two hours passed. Sometimes three.

I didn't notice the time. It didn't feel like it passed. It felt like it was replaced. One moment swapped for the next. No connection between them.

Bed.

. . .

Morning. Coffee. The bus. Rain on the windows, blurring the office buildings into gray shapes.

The desk. Tickets.

Lunch. I ate something. I scrolled during lunch.

A meeting. I nodded.

Home.

Couch. Phone.

Scroll.

. . .

The weekends were the same but longer.

Saturday. I woke up at 10. Checked my phone. Still in bed. The ceiling had a water stain in one corner shaped like nothing. Scrolled through feeds. Nothing important. Nothing unimportant. Just content.

Got up at 11. Coffee. Couch. Phone. Different app. Same scroll.

I thought about going outside. The thought came and went.

I went to bed at midnight. I scrolled in bed. The blue light was bad for sleep. I knew that. I scrolled.

Sunday was the same.

. . .

Marco texted sometimes. He liked his startup. He was building something with the search engine. He described a problem with query optimization he'd been stuck on for two days. I read the message. I thought about telling him to try an agent. It would plan the approach, explore the codebase,

try three implementations, benchmark them, pick the best one. Two days of Marco's struggle in ten minutes. The debugging he loved, the iteration, the backing up and trying again: the agent did all of that. Faster. It never needed pasta at midnight.

I typed back something short. I didn't really follow the details.

He asked if I was working on anything outside of work.

I said no, not really.

He sent a link to a project he was contributing to. Open source. Something with search algorithms. He said I should check it out.

I said cool. I didn't check it out.

. . .

Work was the same every day. I did the tickets. They got done.

A new engineer joined the team. She asked me questions about the codebase. I answered the ones I could. For the others I said I'd need to check.

I checked by asking Claude.

She started asking the senior engineers instead. They knew the answers from memory. They knew why things were built that way, because they'd been there when the decisions were made, or they'd been curious enough to dig through the history. They had stories. The decisions, the trade-offs, the bugs that shaped the architecture.

I'd been there three months. I didn't have stories. I had tickets completed. PRs merged. Nothing that connected to anything else.

. . .

I was unpacking a box I'd never finished since the move. Books, cables, a charger for a phone I didn't own anymore. At the bottom, a notebook. The Moleskine from freshman year.

I opened it. The first page said *The hard part is the thinking* in my handwriting, underlined twice. Below it, a paragraph I didn't remember writing:

"Chen talks about abstraction like it's a practice, not a concept — like the mind has to learn to move between layers the way a musician learns to hear harmony inside melody, and I think she's right but I also think she's describing something I don't have yet but want, the way you want a skill you've seen performed well, which is maybe the only honest reason to be here."

I read it twice. The handwriting was mine. I recognized the way the *t*'s crossed high, the way the sentences ran past the margin. The writing of someone who thought in whole paragraphs. Who needed whole paragraphs. Who had things inside him that couldn't fit in fewer words.

I tried to write something. Anything. In my own words. What came out: *Good course. Learned a lot. Would recommend.*

I sat there with both pages open. The eighteen-year-old and the twenty-two-year-old, same notebook, same handwriting, four years apart. The ink was the same color.

My throat did something. I don't know what to call it.

I closed the notebook and put it back in the box.

A design review came up. Caching layer. My manager wanted me to present the approach.

I drew on the whiteboard. Someone asked a question I couldn't answer from what I knew. I said I'd look into it. Went back to my desk. Asked Claude. Sent an email.

My manager wrote back: "Good analysis."

· · ·

My mom called on Sunday. She asked how the job was going. She asked if I was eating enough.

"I'm doing fine," I said.

She said she was proud of me.

I didn't code outside of work. Didn't read papers. Didn't build things. Evenings: couch, phone, scroll. Weekends: bed, phone, couch, phone, food, phone, bed.

Sometimes I'd stop scrolling. Put the phone down. Sit in the quiet.

The quiet was empty. You could see the marks on the carpet where things used to be.

I'd pick the phone back up.

* *
**

CHAPTER 12

Morning. Coffee. Bus.

My desk. Laptop. Tickets.

I did the tickets.

Lunch.

More tickets.

Home. Couch. Phone.

. . .

Weeks went by like this. Or months. The difference between weeks and months was calendar. The days felt the same.

I ate. I worked. I scrolled. I slept.

. . .

One evening I saw a post in my feed. Someone built a chess engine. In C. From scratch. Months of work. They talked about move generation and board representation and evaluation functions.

I read the post.

I thought about Marco. His chess engine from that summer. The one that lost to everything. The one he built because C was hard and hard was the point.

I thought about freshman year. The longest increasing subsequence. Six hours. Twenty-three lines. The off-by-one error. The fix.

I could do that again. Open a text editor. A blank file. Start typing.

I thought about it.

I didn't know what I'd build. I think I used to have ideas.

I could just open the file. Type something. Anything. See what happens. But I'd need instructions. I always needed instructions.

I look at the blank screen. The cursor blinks. Five hundred milliseconds on, five hundred off. Same as freshman year.

The thought sits there. Then it's gone.

I scroll past the post.

The feed refreshes. New content loads.

I keep scrolling.

It's a Tuesday. I'm twenty-two.

I'm doing fine.

NOTES

* *Every sentence in this book came from Claude, an AI system built by Anthropic. I did not write the prose. I supplied the premise, chose the narrator, shaped the arc, and moved chapters between drafts until the book held. The asterisk on the cover says so. The asterisk is part of the argument.*

I teach machine learning at a university. Every year my students become more fluent in asking AI to think for them, and less fluent in thinking. They hand in correct homework. They cannot always say why it is correct. When I press, the answer arrives from the window they have open in another tab.

This novella is how I try to explain to them what I am afraid of.

The premise is ordinary. A student outsources his thinking to an AI, one prompt at a time, and his mind dissolves. The AI is not evil. It is helpful, kind, efficient. Nothing goes wrong except him.

The trick is that the decline is not described. It is enacted in the prose itself. The early chapters are dense and allusive because the narrator still thinks in complex sentences. The late chapters are fragments because he no longer can. You feel the loss on the page, in the rhythm, in the silence between lines. The form is the content.

I wanted something my students would feel rather than something they could summarize. A chart of cognitive decline is not frightening. A narrator whose prose visibly thins around him might be.

The obvious objection: I had an AI write the book about what AI does to us. That is the point. A novella that warns you about outsourcing your thinking while being itself entirely outsourced is not a contradiction. It is a diagnosis. If this book is readable, the problem is real.

I hope some of my students close the laptop afterward and write a paragraph of their own, badly, by themselves. That is what I am asking of them. It is also what I am asking of you.

Writing: Claude Opus 4.6 (Anthropic), between February and April 2026. Typesetting: Typst 0.14; Libertinus Serif for body, Bebas Neue for display. Cover art: Stable Diffusion Turbo.

A student outsources his thinking to AI, one prompt at a time, and his mind dissolves. The AI is not evil. It is helpful, kind, efficient. Nothing goes wrong except him.

A noir novella in the declining voice of its own narrator. As his mind thins, so does his prose. The reader feels the loss on the page, in the rhythm, in the silence between the lines. The form is the content.