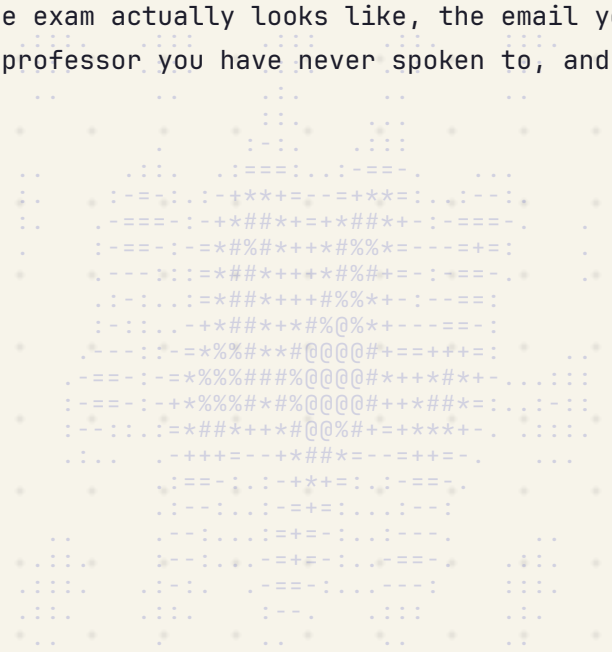
// SECTION 01 · PROMPT PACK

STUDENT CODES

Twenty Claude codes for the parts of school nobody teaches you. The five from the video (/department head, /extension, /attendance, /profiler, /clone) plus fifteen more for exams, papers, group projects, and the money side.

```
> boot opusjake_os
> resource: student-codes
> version: v1.0 · 2026
> status: ready_
```

A code is a nickname for a job you keep re-explaining. Teach Claude the nickname once and one word does the whole thing. These twenty run the parts of school that decide your grade and nobody puts on the syllabus: what the exam actually looks like, the email you are scared to send, the class you missed, the professor you have never spoken to, and your own writing voice.



HOW TO RUN A CODE

There is no backdoor in Claude. A code works because you wrote it. You are compressing a paragraph of instruction into one word the model can recall later, and it holds for the rest of the session.

// TWO WAYS TO FIRE ONE

Inline. Paste the full block, fill the bracketed fields, send. Works in any chat, on any model, on your phone in the hallway. This is the default.

Installed. Paste the pack into a Claude Project's custom instructions once (or into `CLAUDE.md` if you run Claude Code). After that the trigger alone works. Type `/attendance`, attach the slides, done.

// TURN THE RIGHT THINGS ON

- `/department head` and `/profiler` need **web search**. Without it the model guesses, and a guessed past exam is worse than no past exam.
- `/attendance` and `/clone` need **file upload**. Slides, recordings, and your own papers go in as attachments, not as pasted walls of text.
- Everything else runs cold.

// THE RULES

- Paste the block as written. Paraphrasing dilutes it.
- Fill every bracket. Empty brackets are where the model starts inventing.
- One code per message. Two of these fight each other.
- If it drifts back to generic advice, re-fire the code. It re-anchors instantly.

> Every code here tells the model to say when it could not verify something. Keep that line. It is the difference between research and fan fiction.

05 · /DEPARTMENT HEAD

Every course has a paper trail: old finals on the department server, review sessions on YouTube, study guides in a library reserve, five years of syllabi from the same professor. This code goes and finds it, then rebuilds the exam you are about to sit and writes you a mock version of it.

```
/department head: You are reconstructing a course before I sit its exam.
COURSE: [code + full name] · SCHOOL: [university] · TERM: [term]
PROFESSOR: [name]

Search the open web for anything public tied to this course and this
professor: past exams and finals, study guides, review sessions, department
course archives, lecture notes, old syllabi, student threads, library
reserves. Prefer material posted by the school or the professor.

Then give me, in this order:
1. FORMAT: how the exam is actually built. Question types, count, timing,
   open or closed book, point weighting. Rate your confidence per line.
2. RECURRING: topics that appear across multiple past papers, ranked by
   how often they show up.
3. TRAPS: the questions students consistently lose points on, and why.
4. PRACTICE: an eight question mock exam in that exact format, then the
   answer key below a divider.

Cite a link for every factual claim. If you cannot find real material, say
so plainly and build the mock from the syllabus instead of inventing one.
```

Feed it: course code, university, professor, and the syllabus if you have the file.

Use it when: two weeks out, before you decide what to study. Format tells you what to memorize and what to understand.

One rule: if your honor code puts old exams off limits, tell it to skip straight to step 4 and build the mock from the syllabus. A reconstructed practice exam is not a leaked one.

04 · /EXTENSION

Most extension emails fail the same way. They open with an apology, bury the ask in paragraph three, and never quote the policy that already covers the situation. Professors grant extensions that are easy to grant. This code finds the clause, writes the email, and stops you from over explaining.

```
/extension: I need an extension and I want the strongest honest case for it.
SITUATION (all true, do not embellish): [what happened, with dates]
COURSE: [course] · PROFESSOR: [name, how they run the class]
POLICY: [paste the late work and extension section of the syllabus]
I AM ASKING FOR: [the new date]

Do this:
1. Read the policy and quote the exact clause that applies to me. If one
   covers my situation, that is the whole case. If none does, say so and
   build the case on precedent and on what is cheapest for them instead.
2. Tell me what to attach as documentation, and what to leave out.
3. Write the email. Under 150 words including the subject line. Lead with
   the ask and the new date. One sentence of reason. No apology spiral, no
   life story. Close by offering a specific alternative if they say no.
4. Give me the two most likely replies and my one line answer to each.

Invent nothing. Every fact in that email comes from what I gave you.
```

Feed it: the real situation, the syllabus text, the professor's name.

Use it when: before the deadline, not after. The clause that saves you almost always requires you to ask in advance.

03 · /ATTENDANCE

Missing one lecture costs you more than the lecture. It costs the deadline change announced out loud, the topic the professor said would be on the exam, and the thing the slides do not explain. This code turns a recording and a slide deck into the ninety seconds you actually needed to be there for.

```
/attendance: I missed this class. Give me the 90 seconds that matter.
ATTACHED: [lecture recording or transcript, slide deck, announcements]
COURSE + POSITION: [course, today's topic, what is next on the syllabus]

Give me:
1. THE 90 SECONDS: what was taught, in the order it was taught, short
   enough to read out loud in a minute and a half.
2. TESTABLE: the three things most likely to appear on the exam, each
   with the slide number or timestamp it came from.
3. ADMIN: every deadline, format change, or assignment detail said out
   loud but not posted anywhere. Quote it exactly.
4. GAP: what I cannot get from the slides alone and have to ask a human.
5. ONE QUESTION: a single specific question for next class that shows I
   did the reading, not that I was absent.
```

Feed it: the recording or auto transcript, the slides, and a screenshot of the announcements page.

Use it when: the same night. Ask your question at the start of the next class and the absence stops being a story.

02 · /PROFILER

Office hours are usually empty, and the students who go get graded by someone who knows their name. The problem is having nothing to say. This code briefs you on what the professor actually works on, so you walk in with a real question instead of "any tips for the midterm."

```
/profiler: Brief me on this professor before I go to office hours.
PROFESSOR: [name] · DEPARTMENT + SCHOOL: [dept, university]
COURSE: [course] · WHY I AM GOING: [grade, research, recommendation letter]

Use their public professional record only: published papers, faculty or lab
page, course pages and syllabi, recorded talks, department news. No personal
life, no social accounts, no family, nothing they did not publish as part of
their work. If you cannot verify something, leave it out.

Give me:
1. WHAT THEY WORK ON in plain English, and the position they argue that
   others in their field do not.
2. WHAT THAT MEANS FOR THIS COURSE: which parts of the syllabus are their
   real research and which are obligation. They grade the first kind harder
   and care about it more.
3. THREE THINGS TO BRING UP, each tied to something specific they wrote or
   said, with the source link.
4. ONE QUESTION they would enjoy being asked, short enough to ask in 20
   seconds.
5. WHAT NOT TO DO in their office, inferred from how they run the class.
```

Feed it: name, department, school, course, and the syllabus if you have it.

Use it when: the week before you need anything from them. Recommendation letters get written from memory, and you want to be in it.

01 · /CLONE

Every model has a house voice, and everyone can hear it now. This code reads three things you already wrote, reverse engineers how you write, and hands you a style card you can paste at the top of any session. From then on, drafts come back sounding like your other work instead of like a language model.

```
/clone: Learn how I write, then hold that voice for the rest of the session.
ATTACHED: three pieces I wrote myself. [papers, essays, long emails]

Reverse engineer, do not summarize:
1. SENTENCE LENGTH: my average, my range, where I go short on purpose.
2. VOCABULARY: words I reach for, words I never use, my register.
3. STRUCTURE: how I open, how I transition, how I close. Paragraph length.
4. PUNCTUATION AND QUIRKS: dashes, semicolons, parentheses, contractions,
   the moves I repeat without noticing.
5. ARGUMENT HABITS: do I lead with the claim or build to it, how I hedge,
   how I bring in evidence.

Output a STYLE CARD: fifteen rules written in my own voice, specific enough
that another writer could imitate me from the card alone. Add three sample
sentences you generated to prove the match, and name one weakness in my
writing, because copying me faithfully means copying that too.

From now on every draft, edit, and rewrite you give me follows the card.
```

Feed it: three real pieces of your own writing, ideally from different classes.

Use it when: at the start of anything you have to write. Outlines, feedback, and rewrites all come back in your register instead of the model's.

Save the style card somewhere you can find it. The card is the asset, not the session. And use this on work that is yours: detectors flag human writing constantly, so the two things that hold up are a draft that reads like everything else you have written and a document history that shows it being written.

06–10 · STUDY CODES

Five for the week before an exam. Short blocks, same rules.

```
06 · /gap
Quiz me on [topic] one question at a time, hardest first. Do not tell me
the answer until I try. Track what I get wrong and after ten questions
give me the ranked list of what I do not actually know yet.

07 · /feynman
I am going to explain [concept] to you as if you are a smart person from
another department. Interrupt me the moment I use a word I have not
defined or skip a step. At the end, name the exact point my
understanding breaks.

08 · /schedule
My exam is [date] and I have [hours] a week. Build a spaced repetition
plan across these topics: [list]. Weight by how heavily each is graded
and how weak I am at it. Give me a table by day, no motivational filler.

09 · /oneshot
I have [N] hours before this exam and I am not ready. From this material,
give me only what earns the most points per minute: the highest yield
topics, in study order, and what I am consciously giving up. Be honest
about the trade.

10 · /mock
Build a practice exam from these lecture notes: [attached]. Same format
and time limit as the real one. Give me the paper first. Answers, working,
and a per question score band go below a divider so I cannot peek.
```


© OPUSJAKE // CRT V1 · PAGE 08

11–15 · PAPER CODES

Five for the parts of writing a paper that are not writing.

```
11 · /outline
Before I write anything, give me three different structures for this
argument: [thesis]. For each, list the sections and what each has to
prove. Tell me which one is hardest to attack and why.

12 · /rubric
Here is my draft and here is the rubric: [both attached]. Grade it row by
row like the harshest TA in the department. Give the score, the exact
sentence that loses the point, and the smallest fix. No encouragement.

13 · /sources
Check every citation in this draft. For each: does the source exist, does
it say what I claim, and is it the right kind of source for this claim.
Flag anything you cannot verify as UNVERIFIED and never quietly fix it.
List any claim in my paper that has no source at all.

14 · /shrink
Cut this to [N] words without losing an argument. Remove throat clearing,
redundant framing, and any sentence that only restates the one before it.
Show me the cut list so I can veto.

15 · /thesis
My topic is [topic] and my current thesis is [thesis]. Give me five
sharper versions, ordered by how arguable they are. For the top one, name
the strongest counter argument and what evidence I need to survive it.
```


© OPUSJAKE // CRT V1 · PAGE 09

16–20 · CLASS AND MONEY CODES

Five for the admin that quietly costs you money and grades.

16 · /group
Here is the assignment and here are the [N] people in my group with their strengths: [details]. Split it into parts that can be done in parallel, with owners, deadlines two days before the real one, and the exact message I send in the group chat to propose it.

17 · /week
Here are all my syllabi: [attached]. Build one calendar of every deadline, exam, and reading for the term. Flag every week with two or more major deadlines, and tell me which assignment in each pile to start earliest.

18 · /appeal
I am appealing [a grade / a financial aid decision]. Here is the policy and here is what happened: [both]. Find the specific ground my appeal stands on, tell me if I actually have one, and if I do, write the letter in under 250 words. If I do not, say that instead of writing it.

19 · /scholarship
I am a [year, major, school, situation] student. Find scholarships, grants, and department awards I am actually eligible for right now, with deadlines and links. Rank by realistic odds, not by prize size. Include the ones inside my own department that nobody applies to.

20 · /textbook
Find every legal free way to read [book, edition]. Library reserve, open access editions, the previous edition and what changed, interlibrary loan, and the publisher's own free trial. Tell me which one gets it to me before Monday.

INSTALL THE PACK

Inline is fine forever. Installing takes two minutes and makes the triggers work on their own.

1. **Open a Claude Project** and name it after the term, not the class. One project holds every course.
2. **Paste the codes into custom instructions.** All twenty fit. Add one line at the top: "These are my codes. When I type one, run the block it names."
3. **Drop your syllabi into the project knowledge.** Now `/week` and `/extension` already know your deadlines and your late policy, and you stop pasting them.
4. **Add each lecture's slides as you go.** By finals, `/mock` and `/gap` are building from the real course instead of general knowledge.
5. **Keep the style card from `/clone` pinned** at the top of the instructions so every draft comes back in your voice.

On your phone the same project syncs, which is the whole point. The class you miss and the deadline you find out about at 11pm both happen away from your desk.

RUN IT TONIGHT

Pick the one that maps to this week. If you have an exam inside two weeks, run `/department head` on that course now, because format changes what you study and you want that answer early. If you owe a paper, run `/clone` on three things you already wrote and save the style card. If you missed a class today, run `/attendance` before the recording expires.

The codes are not the point. The point is that school runs on information that exists in public and nobody hands you: the format of the exam, the clause in the policy, the research your professor cares about, the award in your own department that goes unclaimed. You now have twenty ways to go get it.

> Steal these. Rewrite them in your own words once you know what each one is doing. A code you edited is a code you will actually remember to use.