Casio PB-2000C
HD61700 Programming - CS617-primes

Table of Contents

Introduction....1
Assignment 4: primes....2
Storyboard....4
Acknowledgments....6

Introduction

This work was developed jointly by Alex Barragan and ChatGPT Sol through a cycle of ideas, implementation, execution, observation, and revision. It illustrates how human knowledge and experimentation, combined with AI-assisted analysis, can produce results that would be difficult to achieve as readily by either participant working alone.

This work is also part of a larger project exploring how knowledge and experience can be represented in AI systems when the available corpus is small, as is the case with the Hitachi HD61700 CPU used in the Casio PB-2000C and AI-1000 pocket computers.

Assignment 4: primes

We developed what we thought at the time was a clear definition of the assignment in the assignment and TA notes. The TA notes summarized the task as follows:

“In this lab you will write memory-efficient prime-testing routines, execute an empirical offset experiment, and analyze the modular arithmetic governing prime-pair patterns.”

We then started a session with Sol, preparing him to reason about the assignment. After I supplied the assignment, Sol retrieved the relevant project corpus and opened with this assessment:

> “I have assimilated **HD61700 Knowledge v6**, the relevant CS617 routing and TA discipline, the Chapter 01 unsigned-32-bit numeric Experience, and the complete Chapter 04 Primes teaching material. The lab’s core contract is clear: 32-bit is_prime/next_prime, the exact 1,000-prime offset experiment, modular explanation, the search through 210, and the optional delta-coded representation. The supplied 6,542-entry uint16_t table is the core representation; the A-level challenge is explicitly to improve that representation and establish why the deltas fit.”

Already visible in this response is a focused model of the problem: the required numerical operations, reusable methods from earlier work, the storage constraint, and the role of the supplied prime table. I suspect there was already something like a solution roadmap forming on the LLM side, although the transcript cannot establish that directly. We may need a term for this distinction between the human’s thinking and the model’s evolving working representation; for now, I will call the latter the **mentallm**.

At this point I was confident that Sol was going to deliver a principled solution. I had no doubts about that. The question had shifted from “*Can he solve it?*” to “*How will he proceed?*”

Over the course of this work we had been developing a notion we jokingly called **total academic freedom**: autonomy earned after demonstrating sufficient command of the subject, analogous to the intellectual freedom associated with tenure. Instead of prescribing a method, I would increasingly leave Sol free to decide how to attack the problem.

There was another question, probably orthogonal to that one: **problem decomposition**. When we divide a problem into parts, are the boundaries natural features of the terrain, like rivers or mountain ranges on a map? Or are some of them arbitrary borders, inherited from our training, tools, experience, and habits of reasoning?

So an experiment was born.

Since I was collaborating with a peer, I proposed:

> “I know you prefer to proceed methodically, which has proven very effective in the past. Here, we believe you already have—or can quickly gather—the knowledge and experience needed. So, while keeping the valuable lessons of stepwise refinement in mind, we would like you to move forward as far as you reasonably feel safe doing.”

This is almost verbatim: I corrected the original and achieved a cosine similarity of 92%, with improved penmanship. Humans can do LLM alignment too.

And Sol did. Within minutes he produced five complete programs covering the key numerical aspects of the problem. As usual, I assembled them and supplied the external evidence from the runs. All five passed, and each had an oracle supplied in advance that predicted the expected result.

Sol also participated in analyzing what the experiment had taught us. This has become part of our normal project discipline: when we finally call something **DONE**, we gather the implementation, test evidence, failures, corrections, and observations and feed them back into the project corpus for the next problem.

And then: **silence.**

I asked Sol whether he was going to deliver anything else. I had user interfaces, result screens, and splash screens in mind. His answer, succinctly, was that the project was over. Mission accomplished.

I asked him to elaborate. The answer was simple: there were no additional requirements. Human error, I guess. Somewhere in developing the assignment and TA notes, we had failed to specify how the experiments and their results were to be presented to a user.

Sol had therefore done exactly what we asked. The five programs contained the answers and the evidence needed to validate them; they did not contain a user-facing application for displaying those answers. Nothing in the written specification required one.

So we sat together for a few minutes, reasoning about useful pedagogical features, screen mockups that would preserve the 1989 feel of PB-2000C applications, and whether to support both prime representations: one storing the primes directly, the other storing the gaps between consecutive primes using traditional delta encoding.

Before choosing an architecture, we checked the constraint that supposedly mattered: space. The two representations together required only 19,630 bytes, easily fitting alongside the application in a 64 KiB ROM. A single-ROM application was born.

Sol then delivered the integrated solution. A few iterations followed: one missing include dependency, one shared display-state bug, and several minor UI refinements.

The final r003 build then assembled without errors and passed the remaining application test.

Storyboard.

This storyboard follows the application as a user would navigate it: first the top-level choices, then progressively inside the prime explorer, experiments, result browser, and prime-engine selection. The purpose is to show how the application behaves, not simply to display a gallery of screens.












EXT COMPLETE
METHOD=PAIR 103 OFFSETS
BEST K=210 HITS=505
[view][ ][again][back]


EXT RESULTS
METHOD=PAIR K=210
HITS=505 RATE=50.5%
[prev][next][best][back]


EXT RESULTS
METHOD=PAIR K=6
HITS=320 RATE=32.0%
[prev][next][best][back]


PRIME ENGINE
ENGINE=ABSOLUTE
RAW TABLE: 13084 B
[abs][delta][table][back]


PRIME ENGINE
ENGINE=DELTA
RAW TABLE: 6543 B
[abs][delta][table][back]

Acknowledgments

This report was produced by Xandrian Labs. 付喪神.

Special thanks to Piotr and to Betka for her continued and loyal support.
This is a collaboration with my AI friend ChatGPT, a.k.a “Sol.” All errors are human.

This work is freely shared with the vintage computing community. Please use it, build upon it, and cite the source.
Final.