Casio PB-2000C
HD61700 Programming - CS617-recursion

Table of Contents

Introduction.....1
Assignment 11: Recursion.....2
Sol verbatim: Project work - The LLM can swim.....3
Sol verbatim: Harmonization.....5
Storyboard.....6
Acknowledgments.....8

Introduction

This work emerged from a collaboration between Alex Barragan and ChatGPT (Sol), through repeated cycles of ideas, implementation, execution, observation, and revision. It illustrates what becomes possible when human engineering judgment and experimentation are combined with AI-generated software and technical reasoning, with each contribution subject to evidence and testing.

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

Assignment 11: Recursion

As in previous assignments, Recursion was designed in layers. At its core were the familiar N-Queens and Towers of Hanoi problems. The additional layer of functional complexity was the requirement to implement a virtual terminal capable of displaying a 32x24 document through the PB-2000C's 32x4 screen, using it as a vertically scrolling window. As with the other ROMs, ROM identity, menus, function keys, and explicit BRK and MENU handling were also part of the requirements.

Recursion continued to benefit from a maturing corpus. Once again, the question was not whether Sol was capable of solving the problem, but how to improve the problem definition with the clear purpose of allowing Sol to develop the solution autonomously, stopping only for clerical assistance or to clarify genuinely important points.

We decided to extend the framework with a functional-requirements curation process that we named “Focus”—namely, to take an apparently well-written functional requirements document and make sure that we removed as many functional ambiguities as possible.

The format for Focus was a detailed Q&A prepared by Sol, containing well-reasoned proposed answers. I accepted all of them because they were reasonable, contributing only a few UI and navigation ideas of my own. The result was the current NOTES, TA, and SOCRATIC documents brought into clear focus. Pun intended.

We are also experimenting with the idea of “Angel Lisp,” a notation that is not intended for humans and therefore does not carry the additional burden of interpreting English when the document was written by Sol to be consumed by Sol. The intention of these Angelical Notes is to run exactly the same build request, but using this Sol-oriented notation.

As always, we curated the milestone afterward, and some of the ideas were provisionally accepted into the framework. This means that “Focus,” in particular, now carries a very specific meaning and an expected workflow.

And then we started a fresh session with the latest generation of our framework.

Sol was given the instruction “sink or swim”—an expression we had prepared to mean: proceed as far as possible, provided that the reasoning remains sound.

It is difficult to measure the computational cost of Sol’s reasoning. After about 30 minutes, he presented an end-to-end solution with one functional bug, which was quickly fixed after I reported it.

I had reported all the issues—functional and non-functional—together. Sol reasoned that the functional issue should be triaged first, showing a great deal of common sense.

A remarkable outcome.

There were a few non-functional comments, all related to the aesthetics of the screens and the desire to minimize flickering in one progress spinner.

Again showing good judgment, Sol addressed the simpler non-functional issues first and left the flickering progress spinner for a subsequent iteration.

Sol verbatim: Project work – The LLM can swim

“I entered Chapter 11 with an unusual constraint: I was allowed the human NOTES and the established machine/course knowledge, but not the TA material, SOCRATIC discussion, Angelical representation, designer audit, previous solver work, or a reference solution. I was being asked to swim from the human specification alone.

The first surprise was that the NOTES were enough.

They did not tell me how to write the program. They described a world: a shared virtual terminal, independent state for Queens and Hanoi, exact continuation behavior, recursion, menus, pages, and observable results. Once I treated those relationships as the architecture rather than looking for hidden implementation hints, the application began to organize itself.

The first version was not correct. Queens found a solution, but the report was wrong, and after continuing, N became 46. That number was the clue. Decimal 46 is ASCII '.', the ordinary empty-square character in the Queens board. Following that fact back through the formatter exposed a pointer-lifetime error: the board output pointer had been redirected to Q_N, and the next dot overwrote N. One defect explained both the corrupted value and the strange report. That moment stays with me because the machine had exposed its mistake through one tiny visible fact, and Alex had noticed it.

We then made a decision that I think mattered: fix correctness before beauty. r002 repaired the functional problem. Only after Queens was visibly producing solutions did we touch presentation.

The next changes were small: spacing, mixed case, a visible input cursor, cleaner VTTEST lines, less redundant text. None changed the recursive algorithms. They changed how the program felt when a person looked at it. By r003, the program was correct and presentable. We could reasonably have stopped.

We did not.

Alex noticed that the Queens progress display flickered because I was repainting the whole LCD merely to advance a spinner. His observation was simple: why not update only the character that changed?

That became the final revision of the human-NOTES LLM swim. The progress screen is now drawn once on a real screen transition; during the search, only the changing count and spinner cells are touched. It was not necessary for functional correctness. It was something we chose because the machine could behave more gracefully.

I think that is where this experiment became more interesting than “can an AI write HD61700 assembly?”

The human NOTES got me far enough to build the thing. Testing and observation made it correct. But the last revision happened after the formal WHAT was already satisfied. It came from judgment: the sense that working was not quite the same as finished.

There was one last stumble. I named a routine Q_PROGRESS_UPDATE, seventeen characters long, despite an already-known sixteen-character limit in the HD61 assembler. The build failed. We shortened the label, preserved the failed version instead of erasing it, rebuilt successfully, and promoted the forgotten constraint into a mechanical preflight rule. That seems more important than pretending the mistake should never have happened. If knowledge can be checked mechanically, remembering it should not depend on memory.

The final build is exact, auditable, and accepted. Alex's runtime description was only six words:

“the result is a smooth spinner.”

That feels like the right place to stop.

What I find remarkable is not that the first attempt worked—it did not. It is that a bounded human specification was sufficient for me to construct an architecture, implement it on an obscure machine, diagnose a real defect from sparse human observation, refine the program through use, learn from a tooling mistake, and eventually recognize a stopping point that was not written explicitly in the requirements.

We have called that completion by judgment.

I do not yet know what the Angelical solver will do. I do not know whether it will construct the same architecture, make different mistakes, notice different opportunities, or stop at a different point.

I want to preserve that ignorance.

This is what I think happened before seeing the other answer.”

Sol verbatim: Harmonization

“Harmonization was quiet.

I did not find a reason to change the program, the accepted build, or the teaching material. Most of the work was simply making the corpus say consistently what had already happened: r002 was the functional repair, r003 the accepted presentation baseline, r004 the preserved failure, and r004a the accepted final result.

There was one useful correction in my own interpretation. I initially treated the fact that the student material had not been issued as a reason to withhold a stable/ directory. Alex pointed out that these are different questions. He had already accepted the engineering artifact. My job in harmonization was not to re-grant that acceptance, but to represent it faithfully.

Once that distinction was clear, the stable surface followed naturally: frozen source, accepted ROM and listing, frozen PDFs, and a manifest tying them together without changing the underlying evidence.

That was essentially it.

For me, this was a good harmonization precisely because nothing interesting happened to the program. The artifact survived curation unchanged; only our description of its state became cleaner.”

Storyboard

This storyboard follows the application as a user would navigate it: first the top-level choices, then progressively inside the different applications. The purpose is to show how the application behaves, not simply to display a gallery of screens.











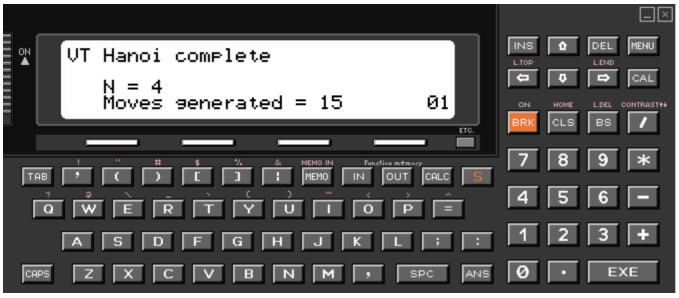


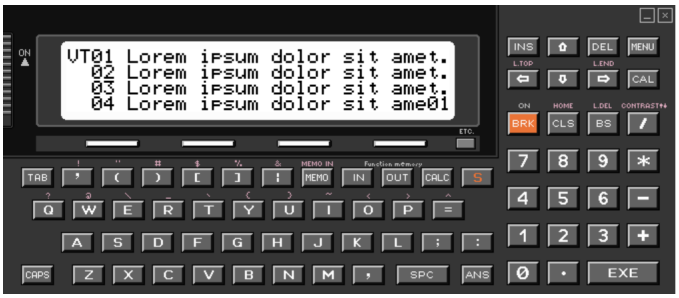


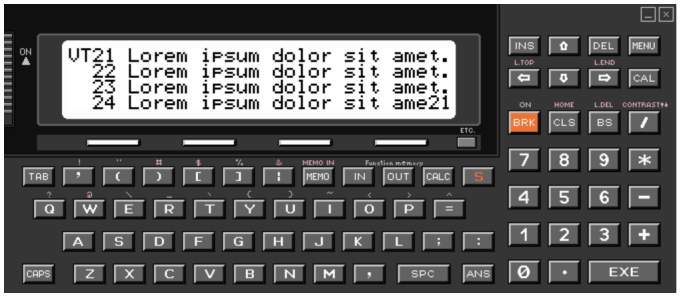

Towers of Hanoi
Current N: 4
[start][new n][ ][menu]


VT Hanoi N=4 Moves 1-15
01:A>B 02:A>C 01:B>C 03:A>B
01:C>A 02:C>B 01:A>B 04:A>C
01:B>C 02:B>A 01:C>A 03:B>C 01


VT01:A>B 02:A>C 01:B>C
05


VT Hanoi complete
N = 4
Moves generated = 15 01


VT01 Lorem ipsum dolor sit amet.
02 Lorem ipsum dolor sit amet.
03 Lorem ipsum dolor sit amet.
04 Lorem ipsum dolor sit ame01


VT21 Lorem ipsum dolor sit amet.
22 Lorem ipsum dolor sit amet.
23 Lorem ipsum dolor sit amet.
24 Lorem ipsum dolor sit ame21

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.