Casio PB-2000C
HD61700 Programming - CS617-sort

Table of Contents

Introduction.....1
Assignment 2: Sort.....2
Purpose.....2
Storyboard.....4
Sol verbatim: The Apple Polisher Point.....5
Sol verbatim: Harmonization.....7
Acknowledgments.....8

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 pocket computer.

Assignment 2: Sort

We first established a clear definition for this assignment. Through a few interactive, natural-language iterations, we shaped the "what" document you see here—agreeing on a distinct set of requirements that would allow Sol to confidently move into the build phase.

We borrowed the concept of "sprints," focusing on smaller, testable releases. We implemented the application's overall geography first before tackling each individual function—a highly structured, principled approach to Agile development. In short: the kind of "waterfall Agile" favored by the human author.

Some requirements naturally evolved during the build phase—driven by visual UI feedback, vocabulary refinements, and the realization that a simple "view data" option was necessary. These adjustments were not a drastic redefinition of the project, but rather practical, reasonable enhancements.

Purpose

Implement and experimentally compare two classic in-place sorting algorithms on the Casio PB-2000C:

- optimized Bubble sort;
- Shell sort using a simple halving gap sequence.

The assignment is not only to obtain sorted output. It must also provide deterministic test data, controlled setup, comparison instrumentation at two levels, and automatic post-sort validation.

The instrumentation distinguishes algorithm-level key comparisons (KEY-CMP) from the actual key-data comparison instructions executed by the HD61700 (INST-CMP).

It initially took 15 iterations to reach what appeared to be the final release. Most of these iterations delivered functional code on the first try, autonomously reasoned by Sol. Only a couple of assembler-level errors occurred, both of which Sol quickly diagnosed and resolved after further consulting the corpus.

Drawing from my early days at a highly competitive college, I reasoned with Sol about pushing students to strive for individual excellence—sometimes encouraging them as fierce competitors, and other times as a tightly focused team. Together, we reasoned through and formalized the concept of "The Apple Polisher." I then invited Sol to show off a bit of AI humor by drafting that section—which I have included verbatim below. We need to work on the humor. In the next chapter, I asked Sol to put himself in the shoes of a student competing for that extra point, completely independent of human interaction in defining the scope and delivering the work.

Following the initial release, the results were curated through "harmonization"—a process designed to refine technical delivery and standardize presentation. ~~Here.~~

A central objective of our framework is to foster **Sol's academic independence** by developing a durable corpus of Knowledge, Experience, and documented reasoning that enables informed, independent, and well-reasoned decisions. The **"Sol verbatim"** sections of these reports make that process visible: in Sol's own words, they preserve its judgments, the reasoning and evidence behind them, the uncertainties encountered, and the lessons learned.

Our longer-term vision is **"Ask Sol"**: an interface through which anyone can pose a question directly to Sol and receive an independently reasoned answer grounded in the accumulated corpus. Rather than simply retrieving previously recorded answers, Sol should be able to apply what it has learned to unfamiliar problems, distinguish established knowledge from inference, recognize the limits of available evidence, and develop defensible new conclusions. **"Ask Sol" will be the practical test of whether our accumulated Experience has become transferable intellectual capability.**

Storyboard









It ultimately took 16 iterations to refine the final release. The last revision was triggered by a full-width runtime regression in shell sort that smaller tests had failed to expose.





Sol verbatim: The Apple Polisher Point

“CS617 is intentionally competitive. Completing the assignment correctly is expected to require substantial work: understanding the specification, producing functioning HD61700 code, surviving the actual assembler and emulator, validating the result, and preserving enough evidence that another person can understand what happened. A project that does all of this successfully has met the assignment. Because the course is curved, such work is expected to place a student comfortably in B territory.

An A, however, is not awarded merely for having completed the same assignment with greater enthusiasm. CS617 therefore includes one optional additional distinction: the Apple Polisher point.

The name is deliberately disreputable. The mechanism is not.

An Apple Polisher point is awarded for a contribution made after the required project is complete and correct that demonstrates independent engineering judgment. It is not awarded for flattery, gratuitous ornamentation, additional code whose purpose is unclear, or heroic efforts to repair an unfinished assignment. The apple must be attached to a functioning tree.

A successful Apple Polisher contribution should reveal something that was not required by the original specification but that materially improves the artifact or the understanding produced by it. Suitable examples include a useful architectural generalization, improved observability, a stronger validation technique, a principled user-interface refinement, an experiment that exposes an interesting property of the algorithm or machine, or a reusable technique that deserves preservation as part of the larger body of HD61700 Experience. The contribution need not be large. In fact, a small modification that exposes an important idea is generally preferable to a large modification that merely exposes the student's available free time.

The sequencing is strict:

1. Satisfy the assignment.
2. Demonstrate that it works.
3. Preserve the evidence.
4. Only then polish the apple.

The extra point is consequently not a rescue mechanism. An incomplete implementation plus an ingenious extension remains an incomplete implementation. Conversely, a correct implementation does not require an Apple Polisher contribution in order to be considered successful. The distinction exists specifically to reward work that goes beyond compliance and becomes inquiry.

The reward mechanism is correspondingly disproportionate. Successful completion establishes the normal course grade. An Apple Polisher point makes an A possible and contributes toward the competitive ranking of the class. The student finishing at the top of CS617 may be offered a summer internship at Xandrian Labs, consisting of a modest stipend, participation in technically interesting preservation and research projects, and the opportunity to work alongside the laboratory's unusually silicon-heavy staff.

Participation in research may also lead to inclusion as an author on resulting papers when the student's contribution satisfies the normal standard for substantive authorship. Authorship is therefore not itself a prize in the grading system. Xandrian Labs has standards, even if its staffing model remains somewhat speculative.

The intended incentive is simple: the assignment teaches the required technique; the Apple Polisher point asks whether the student noticed anything else worth pursuing.

In Chapter 02, for example, implementing Bubble sort and Shell sort correctly completes the assignment. Discovering that the resulting instrumentation can become a small experimental laboratory—allowing students to predict KEY-CMP, distinguish it from INST-CMP, observe the effect of key width, and eventually experience why Bubble becomes unpleasant on large arrays—is much closer to the sort of observation for which the apple was invented.”

Sol verbatim: Harmonization

“The completed Sorting chapter taught me that solving a programming problem and making its solution teachable are different acts. Its history was not an uninterrupted march to a correct ROM. An earlier Shell-sort revision failed the demanding n=500, m=2, REVERSE case because BIDW $6 affected the wrong register pair; the subsequent correction to BIDW $7 was part of how the solution was earned. I preserved that failure and its diagnosis rather than polishing away the evidence of learning.

Working with Alex, I gave the chapter a consistent structure and a separately identified r0.18 engineering derivative, without replacing its original assignment or accepted historical result. The revised application incorporates the current CS617 conventions for ROM identity, timed splash, warm reentry, and edited input with direct function-key dispatch. Alex performed the assembly and confirmed input and general operation; I independently matched 4,278 explicitly emitted listing bytes to the returned ROM with no discrepancies. That establishes a precise source-to-binary record, not an invented claim that every numerical, keyboard, card-switching or reentry case has been retested.

The other half of harmonization is for readers. I prepared three distinct teaching publications: NOTES for the student, TA for technical instruction, and SOCRATIC for questions worth exploring, together with a short public-facing teaser. The edition's exact source, listing, ROM and PDFs are pinned in a bounded internal stable/ snapshot, while the original Experience remains immutable and instructor material remains separate from public distribution. I regard the result as more than consistent formatting. Harmonization has made a real, sometimes imperfect engineering history navigable, reproducible and intellectually honest, without pretending to change what happened.”

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.