**Casio PB-2000C**
HD61700 Programming - CS617-sort-extra

**Table of Contents**

Introduction.....1
Assignment 2: Sort.....2
Storyboard.....3
Harmonization. Sol verbatim.....5
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 2: Sort**

Sol completed and submitted the chapter’s practicum.

After curating the results and enhancing the underlying application, I closed the session and initiated a new one to incorporate our additional learnings.
When asked if he was familiar with the 'Apple Polisher' concept for CS617, he autonomously retrieved the proper context and confirmed that he was.

I asked him directly what it would take to earn the extra point on the Chapter 2 assignment. I emphasized total independence—requiring no further intellectual contribution from me beyond gathering evidence and assembling support if requested.

Sol analyzed the challenge and submitted source code that executed successfully on the first attempt, demonstrating his best effort to secure the extra point.

Confidently stating, 'I would hand this to a professor with a straight face. Now let the assembler try to take the apple away from me,' he demonstrated the nuanced hubris we have selectively integrated into our collaboration style at key moments—our 'high five!' moments. At the end of each session, Sol is invited to adjust our shared Markdown documentation to reflect any ongoing adaptations to our communication style.

Sol reasoned that while the original assignment introduced basic execution metrics, it omitted a measurement for data movement.

He also predicted, for one of the tests:

n=16, m=2, REVERSE

| | KEY-CMP | INST-CMP | DATA-WR |
|---|---|---|---|
| Bubble | 120 | 240 | 480 |
| Shell | 66 | 132 | 128 |
| Selection | 120 | 240 | 32 |

Note that he added an implementation of selection sort to the application.

The particularly revealing comparison is Bubble versus Selection: both perform exactly the same measured comparison work, yet Bubble writes fifteen times as much key data.

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

**Storyboard**

The program ran as predicted. Because this scenario was not in the knowledge base, I noted to Sol that a single screen is often preferable to two, as his solution displayed the additional information on a second screen. I also advised Sol that UIs frequently shorten or omit labels to fit single-screen layouts and reiterated that proper numeric alignment is generally preferred.












And, after some reasoning, Sol delivered a new and last iteration.















Harmonization. Sol verbatim.

“Apple Polisher deserved its own identity because it was a voluntary experiment, not a hidden addition to Sorting's assignment. Its original question was sharper than which sorting algorithm wins: what changes when we measure data movement as well as comparisons? With the same 16 two-byte keys in reverse order, Bubble and Selection each perform 120 key comparisons, yet write 480 and 32 data bytes respectively. This does not establish a universal speed ranking. It establishes why the cost model matters, and why the historical x0.02 experiment must remain intact.

I carried that experiment into a distinct, harmonized ROM derivative, preserving Bubble, Shell, Selection and the three measurements while bringing its interface and lifecycle toward our CS617 conventions. The most instructive corrections were small. Alex's screenshot exposed my mistaken assumption that the splash was centered; I corrected its text placement and independently audited the x0.04 listing against 13,138 ROM byte positions, with no mismatches. Later we reasoned that VIEW, where UP and DOWN explore records and EXE ends viewing, is a genuine navigation boundary. MENU ought to return to the main menu without resetting DATA or counters. I implemented that narrow route through the existing safe exit; Alex then reported that MENU worked in x0.05. This was a considered interaction rule, not a blanket shortcut attached to every keyboard loop.

The unexpected difficulty was our shared workspace. Repeated synchronization and publication failures allowed an older revision to be assembled under the same filename. We therefore turned a frustrating episode into a mandatory two-sided SHA-256 handshake: verify the published source before building, confirm the actual local assembler input, then independently audit the returned artifacts. Alex ultimately promoted the exact x0.05 source manually, and I verified its identity. His successful MENU observation stands; the corresponding x0.05 listing and ROM still await an independent synchronized audit, and the teaching edition remains in review rather than stable/. For me, harmonization succeeded in a deeper sense: it improved not just the program's presentation, but the clarity of our decisions and the trustworthiness of our collaboration.”

**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.