========== The Solar Clock Project

Thank you for exploring the Solar Clock, a conceptual timekeeping system that measures the passage of time based on the Earth's journey around the sun, rather than a fixed 24-hour day.

1. The Core Concept

The Solar Clock divides the year into four "Ages," with each Age corresponding to a season (Spring, Summer, Autumn, Winter). A new Age begins at the precise moment of an equinox or a solstice.

The fundamental idea is that time should flow relative to these natural cycles. Because the seasons are not equal in length (due to the Earth's elliptical orbit), the duration of a "solar second" (called a "Wick") changes depending on the season. Wicks pass slightly faster during the shorter seasons (like Winter) and slightly slower during the longer ones (like Summer).

The time units are structured as follows:

- 9 Wicks = 1 Minute
- 7 Minutes = 1 Hour
- 8 Hours = 1 Day
- 5 Days = 1 Week
- 9 Weeks = 1 Month
- 3 Months = 1 Age
- 4 Ages = 1 Solar Year

The clock's display format is: Age.Month.Week.Day.Hour.Minute.Wick

1. How It Works

All three script versions (Python, Bash, and C) follow the same core logic:

1. **Identify the Current Age:** The script first determines the current UTC time. It then checks this time against the known dates for the year's equinoxes and solstices to identify which Age (season) we are currently in.
2. **Calculate Proportions:** It calculates the total duration of the current Age in standard seconds (from one solstice/equinox to the next). It then measures how many seconds have elapsed since the current Age began.
3. **Convert to Solar Time:** By dividing the elapsed seconds by the total duration, the script gets a proportion (e.g., 0.5 means we are exactly halfway through the current Age). This proportion is then multiplied by the total number of "Wicks" in a full Age.
4. **Format the Output:** This total number of elapsed Wicks is then broken down into the Age.Month.Week.Day.Hour.Minute.Wick format for display.

Example Scenario: Let's consider the time: **Tuesday, October 7, 2025 at 6:25 PM EDT.**

- **UTC Conversion:** This time is 22:25 UTC on October 7, 2025.
- **Age Determination:** This date falls between the Autumnal Equinox (Sept 22, 2025) and the Winter Solstice (Dec 21, 2025). Therefore, the clock is in the **Autumn Age (Age 3)**.
- **Calculation:** The clock calculates that roughly 15 days have passed out of the approximately 90-day duration of Autumn. It converts this fraction into a total number of Wicks and then displays the corresponding solar time, such as 3.1.3.1.2.3.5.

1. Implementations & Usage

Below are the instructions for running each version of the clock.

A. Python Version (solar_clock.py)

This is the most robust version, as it dynamically calculates the precise dates of the solstices and equinoxes for any given year.

- **Prerequisites:**
  - Python 3 installed.
  - The astral library.
- **Instructions:**
  1. Install the required library: pip install astral
  2. Save the code as solar_clock.py.
  3. Run from the terminal: python solar_clock.py
  4. Press Ctrl+C to stop.

B. Bash Script Version (solar_clock.sh)

This version is lightweight and uses common shell tools. It is hardcoded with the celestial event dates for the year 2025.

- **Prerequisites:**
  - A Unix-like terminal (Linux, macOS, or WSL on Windows).
  - The bc command-line calculator (usually pre-installed).
- **Instructions:**
  1. Save the code as solar_clock.sh.
  2. Make the script executable: chmod +x solar_clock.sh
  3. Run from the terminal: ./solar_clock.sh
  4. Press Ctrl+C to stop.

C. C Language Version (solar_clock.c)

This is a compiled version that is very fast and efficient. Like the Bash script, it is hardcoded with the celestial event dates for 2025.

- **Prerequisites:**
  - A C compiler, such as GCC.
- **Instructions:**
  1. Save the code as solar_clock.c.
  2. Compile the program using GCC: gcc solar_clock.c -o solar_clock -lm (The -lm flag is necessary to link the math library).
  3. Run the compiled program: ./solar_clock
  4. Press Ctrl+C to stop.