Problem Title: Warehouse Inventory Discrepancy (abs)

- **Input**: physical_count (Integer), system_count (Integer)
- **Scenario**: A warehouse manager needs to find the total units missing or over-counted. Regardless of whether the system shows more or less than the shelf, the manager needs the absolute difference as a positive "discrepancy value."
- **Output**: A single positive numeric value representing the magnitude of the error.

Problem Title: Event Seating Planner (ceiling)

- **Input**: total_guests (Integer), table_capacity (Integer)
- **Scenario**: You are organizing a wedding. If you have 101 guests and each table fits 10, you cannot have 10.1 tables; you must provide enough tables for everyone.
- **Output**: The total number of tables required (must be a whole number).

Problem Title: Grocery Budget Truncation (floor)

- **Input**: account_balance (Numeric), item_price (Numeric)
- **Scenario**: A shopper wants to know the maximum number of full units of a product they can buy. Even if they can afford 5.99 units, the store only sells whole units.
- **Output**: The maximum whole number of items affordable.

Problem Title: Lab Result Precision (signif)

- **Input**: chemical_concentration (Numeric/Scientific)
- **Scenario**: A laboratory produces results ranging from 0.00003421 to 1250.4. To maintain scientific integrity, all reports must be published using exactly 3 significant figures, regardless of the decimal position.
- **Output**: The numeric value adjusted to 3 significant digits.

Problem Title: Financial Statement Beautifier (formatC)

- **Input**: transaction_amounts (Numeric Vector)
- **Scenario**: A bank is generating a text-based monthly statement. All numbers must:
    1. Always show exactly 2 decimal places (even for whole numbers like $50.00).
    2. Include a comma as a thousands separator.
    3. Be padded with leading spaces to align perfectly in a column of width 12.
- **Output**: A character vector of formatted strings.

Problem Title: Bacteria Growth Modeling (exp / log)

- **Input**: initial_count (Numeric), growth_rate (Numeric), time_hours (Numeric)
- **Scenario**: A scientist is modeling "Continuous Growth." Use the constant e (natural exponential) to calculate the final population. Then, calculate the "Doubling Time" by finding the natural logarithm of 2 divided by the growth rate.
- **Output**: Two values: The projected population and the time required to double.

Problem Title: Stock Market Trend Smoothing (stats::filter)

- **Input**: daily_closing_prices (Numeric Vector)

- **Scenario**: An analyst wants to remove daily "noise" from a stock's performance. Apply a 5-day backward-looking moving average.
    - *Rule:* Each point in the output should be the average of the current day and the four preceding days.
    - *Constraint:* The first 4 days should result in NA as they lack sufficient history.
- **Output**: A numeric vector (or time-series object) representing the smoothed trendline.

Problem Title: Cumulative Revenue Milestone (cumsum)

- **Input**: daily_sales_revenue (Numeric Vector)
- **Scenario**: A startup wants to identify exactly which day they hit their first $1,000,000 in total sales.
    1. Calculate a running total of revenue.
    2. Identify the index (day) where that running total first meets or exceeds the target.
- **Output**: A vector of the running total and the integer index of the milestone day.

Problem Title: The Precision Trap (Floating Point)

- **Input**: val1 (Calculated as $0.1 + 0.2$), val2 (Target value 0.3)
- **Scenario**: In computer science, $0.1 + 0.2$ does not always exactly equal 0.3 due to binary precision. Create a logical check that returns TRUE if the values are "close enough" within a tolerance of $1e-10$, even if they aren't identical.
- **Output**: A Boolean (TRUE/FALSE) confirming if the numbers are effectively equal.

Problem Title: Daily Temperature Volatility (diff)

- **Input**: hourly_temps (Numeric Vector)
- **Scenario**: A meteorologist is studying how fast weather changes. Instead of the temperatures themselves, they need the **change** between each hour. If the temp goes from 20 to 22, the change is +2.
- **Output**: A numeric vector representing the step-by-step difference between consecutive elements. (Note: The output will be 1 element shorter than the input).

Problem Title: Sensor Outlier Detection (range)

- **Input**: sensor_readings (Numeric Vector)
- **Scenario**: A factory sensor is being calibrated. The engineer needs to know the total "spread" of the data (the minimum and maximum recorded values) to see if any reading exceeded safety limits.
- **Output**: A numeric vector of length 2 showing the lowest and highest values.

Problem Title: Tax Bracket Categorization (cut)

- **Input**: incomes (Numeric Vector), breaks (Numeric Vector)
- **Scenario**: A government agency needs to sort citizens into "Low," "Medium," and "High" earners based on specific income thresholds.
    - *Rule:* Use thresholds [0, 30k, 70k, 150k].
    - *Action:* Transform the continuous income numbers into a categorical factor with descriptive labels.
- **Output**: A factor vector where each number is replaced by its corresponding bracket label.

Problem Title: Shipping Container Weight Limit (pmin)

- **Input**: actual_weights (Numeric Vector), max_allowable (Single Numeric Value)
- **Scenario**: A logistics company is loading trucks. For safety regulations, no recorded weight in the report can exceed the legal limit. If a weight is over the limit, it must be "capped" (replaced) by the maximum allowable value.
- **Output**: A numeric vector where all values exceeding the limit are replaced by the limit itself, but lower values remain unchanged.

Problem Title: Exam Score Standardization (scale)

- **Input**: test_scores (Numeric Vector)
- **Scenario**: Two different classes took two different versions of an exam. To compare them fairly, you must calculate the **Z-score** for each student.
    - *Logic:* Subtract the mean of the scores from each individual score, then divide the result by the standard deviation.
    - *Constraint:* This centers the data at 0 and expresses the score in "standard deviations" away from the average.
- **Output**: A numeric vector of standardized scores.

Problem Title: The Weighted Grade Point Average (Weighted Mean)

- **Input**: grades (Numeric Vector 0-100), credits (Numeric Vector 1-5)
- **Scenario**: A university calculates a student's GPA. However, a 5-credit "Physics" class should influence the average more than a 1-credit "Physical Education" class.
    - *Formula:* $\sum (grade \times credit) / \sum (credits)$
    - *Task:* Calculate the final GPA without simply averaging the grades column.
- **Output**: A single numeric value representing the weighted average.

Problem Title: Logarithmic Scaling for Data Visualization (log10)

- **Input**: city_populations (Numeric Vector ranging from 1,000 to 10,000,000)
- **Scenario**: You are creating a chart. Because the populations vary so much, the small cities disappear on the graph. You need to transform the data to a **Log10 scale** to compress the distance between the small and large values.
    - *Validation:* Ensure the function handles a population of 0 (if it exists) without returning an error or "Inf."
- **Output**: A numeric vector of log-transformed values.