**Technical Design**

**Name:** Livia Augusto Razera

**Date Created:** June 15, 2025

**Program Description:**

This program prompts users to enter a list of their monthly specified monthly expenses, including type and amount. Then it calculates and displays the total amount and identifies the highest and lowest expenses.

**Functions used in the Program (list in order as they are called):**

1. **Function Name:** get\_expenses

**Description:** Prompts the user repeatedly to input the type of expense and its amount until the user types 'done'. Collects all expenses as tuples of (expense\_type, amount) in a list.

**Parameters:** none.

**Variables:**

expenses (list of tuples): Stores the entered expense type and amount pairs.

expense\_type (string): Stores the current expense type input from the user.

amount (float): Stores the amount associated with the current expense type.

**Logical Steps:**

1. Initialize an empty list expenses.
2. Print instructions for the user.
3. Loop indefinitely asking the user for expense type.
4. If the user enters 'done' (case insensitive), break the loop.
5. Otherwise, prompt the user to enter the expense amount.
6. Attempt to convert the amount to a float; if invalid, prompt again.
7. Append the tuple (expense\_type, amount) to expenses.
8. Return the list expenses after the loop ends.

**Returns**: A list of tuples, each tuple containing (expense\_type: str, amount: float).

**2. Function Name:** main()

**Description:**  Calls the other function, then uses reduce with lambda functions to get total, highest, and lowest expense. It also displays the results.

**Parameters:** none

**Variables:**

expenses (list of tuples): The list returned from get\_expenses.

total (float): Sum of all expense amounts.

highest (tuple): Expense tuple with the highest amount.

lowest (tuple): Expense tuple with the lowest amount.

**Logical Steps:**

1. Call get\_expenses() and assign its return value to expenses.
2. If expenses is empty, print a message and exit the function.
3. Use reduce() and a lambda function to sum all expense amounts for total.
4. Use reduce() and a lambda function to find the highest expense tuple.
5. Use reduce() and a lambda function to find the lowest expense tuple.
6. Print the total expense, highest expense (with type and amount), and lowest expense (with type and amount).

**Returns:** none

**Logical Steps:**

1. Program starts and calls main().
2. main() calls get\_expenses() to collect user input.
3. User inputs expense types and amounts until typing 'done'.
4. get\_expenses() returns the collected expenses list to main().
5. main() analyzes expenses with reduce() to compute totals and extremes.
6. Results are printed to the console for the user.

**Link to your repository:** <https://github.com/liviaar/COP2373.git>

**Output Screenshot:**

**A screen shot of a computer

AI-generated content may be incorrect.**