**Technical Design Document**

**Name:** Henry Huitema

**Date Created:** July 29th, 2025

**Program Description:**

This program collects a list of monthly expenses from the user, and uses the reduce method to find and display the highest expense, lowest expense, and total expenses.

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

1. **Function Name:** getHigher

**Description:** This function takes two numerical values and returns whichever is greater. If both are equal, it returns the second argument by default.

**Parameters:**

a (float) – First number to be compared.

b (float) – Second number to be compared.

**Variables:**

None used aside from parameters.

**Logical Steps:**

1. If a is greater than b, return a.
2. If a is less than or equal to b, return b.

**Returns:** a or b.

2. **Function Name:** getLower

**Description:** This function takes two numerical values and returns whichever is lower. If both are equal, it returns the second argument by default.

**Parameters:**

a (float) – First number to be compared.

b (float) – Second number to be compared.

**Variables:**

None used aside from parameters.

**Logical Steps:**

1. If a is less than b, return a.
2. If a is greater than or equal to b, return b.

**Returns:** a or b.

3. **Function Name:** keyFromValue

**Description:** This function takes a value and dictionary. If a value is in the dictionary, it returns the key associated with said value.

**Parameters:**

tar – Value used to search for a key.

dict (dictionary) – Dictionary to be searched.

**Variables:** None used aside from parameters.

**Logical Steps:**

1. Iterate through each item in dictionary using a for loop. If the tar parameter matches a value in the dictionary, return its corresponding key.
2. If the loop completes without returning a key, return an empty string.

**Returns:** Key associated with tar value. If no match is found, empty string.

4. **Function Name:** main

**Description:** The main function takes a user-supplied set of expenses and displays the highest expense, lowest expense, and total expenses.

**Parameters:** This function takes no parameters.

**Variables:**

keepGoing (string) – Used to determine whether or not to end the while loop used to take expenses. If keepGoing is any value except ‘y’, the loop ends.

expenses (dictionary) – Dictionary of expense types and corresponding amounts of money.

expenseType (string) – Type of expense input by the user.

expenseAmount (float) – Amount of money input by the user.

lowestExpense (float) – Lowest expense in expenses dictionary.

highestExpense (float) – Highest expense in expenses dictionary.

**Logical Steps:**

1. Using a while loop, collect expense types and amounts until user indicates they would like to stop and store expenses in a dictionary.
2. Use getLower on the values of the expense dictionary within the functools.reduce method to find the lowest expense. Store it in lowestExpense variable.
3. Use getHigher on the values of the expense dictionary within the functools.reduce method to find the highest expense. Store it in highestExpense variable.
4. Within a formatted string, call keyFromValue with lowestExpense and expenses as arguments to find the expense type corresponding to lowestExpense, and print it alongside lowestExpense.
5. Within a formatted string, call keyFromValue with highestExpense and expenses as arguments to find the expense type corresponding to highestExpense, and print it alongside highestExpense.
6. Calculate total expenses using a lambda function on the values of the expenses dictionary to sum every value, then display the result within a formatted string.

**Returns:** This function does not return anything.

**Link to your repository:** https://github.com/HenryH-SCF/COP2373

**Output Screenshot: (make sure big enough so I can see)**

