**Technical Design Document Template**

**Name:** Noah Muncie

**Date Created:** 02/16/2025

**Program Description:**

This program is designed to get user input of their expenses and return their total expenses, highest expense, and lowest expense.

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

**Function Name:** get\_expenses()  
**Description:** This function asks the user to input their expenses, including the type and the amount. It loops until the user types "done" and stores each expense as a tuple in a list.  
**Parameters:**

* None  
  **Variables:**
* expenses (list): A list that holds the expense data, where each entry is a tuple consisting of the type of expense (string) and the amount (float).
* expense\_type (string): Holds the type of the expense, entered by the user.
* expense\_amount (float): Holds the amount of the expense, entered by the user.  
  **Logical Steps:**

1. Initialize an empty list expenses to store the user input.
2. Start a loop to prompt the user for input.
3. If the user types "done", break out of the loop.
4. For other input, ask for the amount and convert it to a float. If the input is invalid, ask again.
5. Append a tuple containing the expense\_type and expense\_amount to the expenses list.  
   **Returns:**

* expenses (list): A list of tuples, each containing the type of expense (string) and the amount (float).

**Function Name:** total\_expense(expenses)  
**Description:** This function calculates the total amount of all expenses by summing the values in the expenses list using the reduce function.  
**Parameters:**

* expenses (list): The list of expenses, where each element is a tuple containing the expense type and the amount.  
  **Variables:**
* total (float): Holds the accumulated sum of the expenses as the reduce function processes each expense.  
  **Logical Steps:**

1. Use reduce to iterate over the expenses list.
2. For each expense, add its amount (expense[1]) to a running total.
3. Start the accumulation with an initial value of 0.  
   **Returns:**

* total (float): The total sum of all expenses.

**Function Name:** highest\_expense(expenses)  
**Description:** This function finds the expense with the highest amount in the expenses list. It uses the reduce function to compare each expense and keep track of the highest one.  
**Parameters:**

* expenses (list): The list of expenses, where each element is a tuple containing the expense type and the amount.  
  **Variables:**
* highest (tuple): A tuple that holds the current highest expense found during iteration. It stores both the type and amount of the highest expense.  
  **Logical Steps:**

1. Use reduce to iterate over the expenses list.
2. For each expense, check if its amount (expense[1]) is greater than the current highest expense.
3. If so, update highest with the current expense; otherwise, keep the previous highest.  
   **Returns:**

* highest (tuple): A tuple containing the type of the highest expense and its amount.

**Function Name:** lowest\_expense(expenses)  
**Description:** This function finds the expense with the lowest amount in the expenses list. It uses the reduce function to compare each expense and track the lowest one.  
**Parameters:**

* expenses (list): The list of expenses, where each element is a tuple containing the expense type and the amount.  
  **Variables:**
* lowest (tuple): A tuple that holds the current lowest expense found during iteration. It stores both the type and amount of the lowest expense.  
  **Logical Steps:**

1. Use reduce to iterate over the expenses list.
2. For each expense, check if its amount (expense[1]) is less than the current lowest expense.
3. If so, update lowest with the current expense; otherwise, keep the previous lowest.  
   **Returns:**

* lowest (tuple): A tuple containing the type of the lowest expense and its amount.

**Logical Steps:**

1. Call the get\_expenses() function to prompt the user for their expenses and return a list of expenses.
2. Check if the expenses list is empty. If no expenses are entered, print "No expenses entered" and stop the program.
3. Call the total\_expense(expenses) function to calculate the total sum of the expenses.
4. Call the highest\_expense(expenses) function to find the highest expense in the list.
5. Call the lowest\_expense(expenses) function to find the lowest expense in the list.
6. Display the results: Print the total amount of expenses, the type and amount of the highest expense, and the type and amount of the lowest expense.

**Link to your repository:** <https://github.com/NMHero1/COP2373>

