Allowance and Expense Tracker – Documentation

I. Project Overview

The **Allowance and Expense Tracker** is a command-line program designed to help users manage their weekly allowance and track their expenses. This tool is particularly useful for individuals who want to stay on top of their spending and make sure they do not exceed their weekly budget.

II. Features and Explanation

i. Setting Weekly Allowance

```
void setAllowance() {
    cout << "Enter your weekly allowance: ";
    cin >> allowance;

    if (allowance < 0) { // Validate that the allowance is not negative
        cout << "Allowance cannot be negative. Please try again." << endl;
        return;
    }

    cout << "Allowance set to PHP " << fixed << setprecision(2) << allowance << "." << endl;
    checkAllowance(); // Warn the user if allowance is low
}
```

a. Users can input their weekly allowance, which is the starting balance.

b. The program will ensure the allowance is non-negative. If the user enters a negative value, they will be prompted to enter a valid amount.

ii. Adding an Expense (Create)

```
void addExpense() {
    if (allowance <= 0) { // Ensure allowance is set before adding an expense
        cout << "Please set your allowance first." << endl;
        return;
    }

    Expense exp; // Create a new expense
    exp.id = nextId++; // Assign a unique ID to the expense

    cout << "Enter expense name: ";
    cin.ignore(); // Clear the input buffer
    getline(cin, exp.name);

    cout << "Enter expense amount: ";
    cin >> exp.amount;
```

```
    if (exp.amount < 0) { // Ensure expense amount is valid
        cout << "Expense amount cannot be negative." << endl;
        return;
    }

    if (exp.amount > allowance) { // Check if there's enough allowance for the expense
        cout << "Insufficient allowance for this expense." << endl;
        return;
    }

    allowance -= exp.amount; // Deduct expense amount from allowance
    expenses.push_back(exp); // Add the expense to the list
    cout << "Expense added successfully! Remaining allowance: PHP "
         << fixed << setprecision(2) << allowance << "." << endl;
    checkAllowance();
}
```

a. Users can add an expense, entering both a name/description and the amount spent.

b. The program checks that the expense does not exceed the available allowance. If it does, the user will be warned.

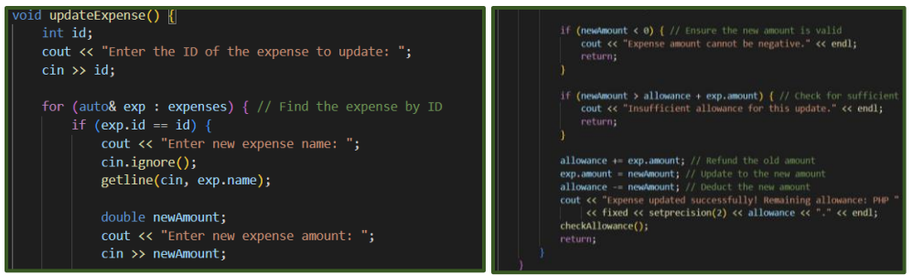
iii. Viewing Expenses (Read)

```
void viewExpenses() {
    if (expenses.empty()) { // Check if there are no expenses
        cout << "No expenses recorded yet." << endl;
        return;
    }

    cout << endl << "=== Expenses ===" << endl;
    cout << "ID   Name                Amount" << endl;
    cout << "-------------------------------" << endl;

    for (const auto& exp : expenses) { // Loop through all expenses and display them
        cout << setw(4) << exp.id << "   " << setw(18) << left << exp.name
             << "PHP " << fixed << setprecision(2) << exp.amount << endl;
    }

    cout << "-------------------------------" << endl;
    cout << "Remaining allowance: PHP " << fixed << setprecision(2) << allowance << "." << endl;
    checkAllowance();
}
```

a. All recorded expenses are displayed with their ID, name, and amount.

b. The remaining allowance is updated and displayed after each expense.

iv. Updating an Expense (Update)



```
void updateExpense() {
    int id;
    cout << "Enter the ID of the expense to update: ";
    cin >> id;

    for (auto& exp : expenses) { // Find the expense by ID
        if (exp.id == id) {
            cout << "Enter new expense name: ";
            cin.ignore();
            getline(cin, exp.name);

            double newAmount;
            cout << "Enter new expense amount: ";
            cin >> newAmount;

            if (newAmount < 0) { // Ensure the new amount is valid
                cout << "Expense amount cannot be negative." << endl;
                return;
            }

            if (newAmount > allowance + exp.amount) { // Check for sufficient
                cout << "Insufficient allowance for this update." << endl;
                return;
            }

            allowance += exp.amount; // Refund the old amount
            exp.amount = newAmount; // Update to the new amount
            allowance -= newAmount; // Deduct the new amount
            cout << "Expense updated successfully! Remaining allowance: PHP "
                 << fixed << setprecision(2) << allowance << "." << endl;
            checkAllowance();
            return;
        }
    }
```

a. Expenses can be updated by their ID. Users can change both the name and the amount.

b. The new expense amount must not exceed the remaining allowance, and if the amount is negative, the user is notified.

v. Deleting an Expense (Delete)

```
void deleteExpense() {
    int id;
    cout << "Enter the ID of the expense to delete: ";
    cin >> id;

    for (auto it = expenses.begin(); it != expenses.end(); ++it) { // Find the expense by ID
        if (it->id == id) {
            allowance += it->amount; // Refund the deleted expense amount
            expenses.erase(it); // Remove the expense from the list
            cout << "Expense deleted successfully! Remaining allowance: PHP "
                 << fixed << setprecision(2) << allowance << "." << endl;
            checkAllowance();
            return;
        }
    }

    cout << "Expense with ID " << id << " not found." << endl;
}
```

a. Users can delete an expense by entering its ID. The expense will be removed from the list, and the amount will be refunded to the allowance.

vi. Allowance Status Check

```
void checkAllowance() {
    static double initialAllowance = allowance; // Save the initial allowance
    const double warningThreshold = 0.2 * initialAllowance; // 20% of the initial allowance

    if (allowance <= 0) { // No allowance left
        cout << "WARNING: Your allowance is completely used up. Be careful with your spending!" << endl;
    } else if (allowance < warningThreshold) { // Allowance is low
        cout << "Reminder: Your remaining allowance is low. Spend wisely!" << endl;
    }
}
```

a. The program warns users when their remaining allowance is low (below 20%) or when it is completely used up, reminding them to spend wisely.

vii. Error Handling

a. If the user enters an invalid input (e.g., a negative expense amount), the program will notify them and prompt for correct input.

III. Instructions for Using the Program

i. Step 1: Set Your Weekly Allowance

```
Enter your choice: 1
Enter your weekly allowance: 1500
Allowance set to PHP 1500.00.
```

a. Select **1** to set your weekly allowance.

b. Enter a non-negative value. The allowance will be stored, and the program will display the set value.

ii. Step 2: Add an Expense

```
Enter your choice: 2
Enter expense name: breakfast
Enter expense amount: 100
Expense added successfully! Remaining allowance: PHP 1400.00.
```

a. Select **2** to add an expense.

b. Enter the expense name and amount. If the amount exceeds the available allowance, the program will display a warning and prevent the expense from being added.

iii. Step 3: View Your Expenses

```
Enter your choice: 3

=== Expenses ===
ID   Name                Amount
-------------------------------
   1   breakfast         PHP 100.00
-------------------------------
Remaining allowance: PHP 1400.00.
```

a. Select **3** to view all recorded expenses.

b. The program will display each expense's ID, name, and amount, as well as the remaining allowance.

iv. Step 4: Update an Expense

```
Enter your choice: 4
Enter the ID of the expense to update: 1
Enter new expense name: breakfast
Enter new expense amount: 200
Expense updated successfully! Remaining allowance: PHP 1300.00.
```

a. Select **4** to update an existing expense.

b. Enter the expense ID, then update the name and/or amount. The updated amount must not exceed your available allowance.

v. Step 5: Delete an Expense

```
Enter your choice: 5
Enter the ID of the expense to delete: 1
Expense deleted successfully! Remaining allowance: PHP 1500.00.
```

a. Select **5** to delete an expense by ID.

b. The amount of the deleted expense will be refunded to your allowance.

vi. Step 6: Exit the Program

```
Enter your choice: 6
Exiting program. Goodbye!
```

a. Select **6** to exit the program. The program will terminate with a goodbye message.