Assignment 8: Functional Programming in Python

Scenario: ECommerce Data Analysis

Imagine you work as a data analyst for an e-commerce company. Your team has a list of customer orders, each containing information about items purchased, quantity, price, and customer information. The company wants you to generate insights about the data using a **map, reduce, and filter functions**. The data set is given below:

```
orders = [
  {'order_id': 1, 'customer': 'Alice', 'items': [
    {'name': 'Laptop', 'price': 1200, 'quantity': 1},
    {'name': 'Mouse', 'price': 25, 'quantity': 2}
  ]},
  {'order_id': 2, 'customer': 'Bob', 'items': [
    {'name': 'Laptop', 'price': 1200, 'quantity': 1}
  ]},
  {'order_id': 3, 'customer': 'Charlie', 'items': [
    {'name': 'Monitor', 'price': 300, 'quantity': 1},
    {'name': 'Keyboard', 'price': 70, 'quantity': 1}
  ]},
  {'order_id': 4, 'customer': 'David', 'items': [
    {'name': 'Laptop', 'price': 1200, 'quantity': 2},
    {'name': 'Keyboard', 'price': 70, 'quantity': 1}
  ]}
]
```

1. Identify Customers with Large Orders

Using the ecommerce scenario, filter for orders with more than three items and return a list of customer names.

Expected Output:

A list of customer names who ordered more than three items.

Assignment 8: Functional Programming in Python

2. Calculate Total Cost for Each Item Type

Write a function that calculates the total revenue generated by each item type across all orders. Use `map` to extract the relevant item details and `reduce` to aggregate total costs by item type.

Expected Output:

{'Laptop': 4800, 'Mouse': 50, 'Monitor': 300, 'Keyboard': 140}

3. Identify Unique Products Purchased by Each Customer

For each customer, determine the unique products they purchased. Use `map` to iterate over each customer's order and `filter` to remove duplicates.

Expected Output:

A dictionary where the keys are customer names and values are lists of unique items they bought.

4. Calculate the Average Quantity of All Items Sold

Calculate the average quantity of each item sold across all orders. Use `map` to get the quantity of each item and `reduce` to calculate the total and average quantities.

Expected Output:

1.75 (if total quantity of all items is 7 and there are 4 unique items)

5. Find Customers Who Ordered Only One Type of Item

Using `filter`, identify and return a list of customer names who ordered only a single type of item (regardless of quantity).

Expected Output:

A list of customers who ordered just one type of product.

6. Count Total Units of Each Product Sold

Using a combination of `map` and `reduce`, count the total units sold for each product across all orders.

Expected Output:

{'Laptop': 4, 'Mouse': 2, 'Monitor': 1, 'Keyboard': 2}

Assignment 8: Functional Programming in Python

7. Filter Orders Made by VIP Customers

Assume you have a list of VIP customers. Using `filter`, return only the orders made by customers on the VIP list.

Input:

vip_customers = ['Alice', 'David'] **Expected Output:**

Only the orders made by `Alice` and `David`.

8. Identify Orders with HighValue Items

Identify orders that contain items priced above $1000. Use `filter` to get the relevant items and `map` to return the order IDs.

Expected Output:

A list of order IDs containing items priced above $1000.

9. Calculate Total Cost of Orders Using Custom Discounts

Each item has a discount percentage based on the product type. Use `map` to apply discounts to each item’s price in the order and `reduce` to calculate the final cost of the order after discounts.

Discounts Example:

discounts = {'Laptop': 0.1, 'Monitor': 0.05, 'Mouse': 0, 'Keyboard': 0.15}

Expected Output:

A list of final order prices after applying discounts to individual items.

10. Find Customers with All Orders Containing Laptops

Using `filter`, find customers who have purchased a `Laptop` in every one of their orders.

Expected Output:

A list of customer names who have laptops in all their orders.

Assignment 8: Functional Programming in Python

11. Calculate the Maximum Quantity of Any Single Product Ordered

Using `map` and `reduce`, determine the highest quantity of any single product ordered across all orders.

Expected Output:

For instance, if one order has a quantity of 4 Laptops, the output should be `4`.

12. Filter Customers with Orders Below a Certain Total

For each customer, filter out orders where the total cost is less than $500.

Expected Output:

A dictionary of customers with lists of orders, each having a total cost of at least $500.

13. Find the Total Number of Unique Products Purchased

Use `map` to extract all product names from each order and `reduce` to count the unique products purchased across all orders.

Expected Output:

The count of unique products purchased across all orders.

14. Create a List of Items Ordered by Each Customer

For each order, create a list of items purchased (just the item names) for each customer.

Expected Output:

A dictionary where the keys are customer names and values are lists of items they ordered.

15. Calculate Total Price of Items with a Given Discount Applied

Assume a 10% discount on all items. Use `map` to apply the discount to the price of each item and `reduce` to calculate the total discounted price across all orders.

Expected Output:

The total discounted price for all items in all orders.