⍰ Low-Level Design (LLD) for Shopping Bill Calculator

Difficulty Level: Easy | Total Marks: 10
Standards Followed: 4 Functions | 3 Visible Test Cases | 2 Hidden Test Cases

---

□ Concepts Tested

□ NumPy Arrays and Vectorized Operations
□ Arithmetic and Aggregation
□ Conditional Filtering
□ Class-Based Programming

□ Problem Statement

Develop a billing calculator system that takes product prices, quantities, applies a discount, and filters expensive items based on a threshold.

It should use NumPy arrays for all calculations and should support:
• Creating an array of prices
• Computing the total cost
• Applying a discount to every item
• Filtering prices that exceed a threshold

□ Operations

1. Create Price Array
   □ Converts a list of prices into a NumPy array
   □ Function Prototype:
   def create_price_array(self, price_list: list)
   □ Example Input:
   create_price_array([100, 200, 300])
   □ Expected Output:
   [100. 200. 300.]

---

2. Calculate Total Cost
   □ Returns total cost using prices × quantities
   □ Function Prototype:
   def calculate_total_cost(self, prices: np.ndarray, quantities: np.ndarray)
   □ Example Input:
   calculate_total_cost([100, 200], [2, 1])

□ Expected Output:
400.0

---

3. Apply Discount
   □ Applies a percentage discount to all prices
   □ Function Prototype:
   def apply_discount(self, cost_array: np.ndarray, discount_percent: float)
   □ Example Input:
   apply_discount([100, 200], 10)
   □ Expected Output:
   [90.0 180.0]

---

4. Filter Items Above Threshold
   □ Returns values from cost array above the given threshold
   □ Function Prototype:
   def filter_items_above_threshold(self, cost_array: np.ndarray, threshold: float)
   □ Example Input:
   filter_items_above_threshold([90.0, 180.0], 100)
   □ Expected Output:
   [180.0]

□ Implementation Code

```
python
CopyEdit
import numpy as np

class ShoppingBill:
    def create_price_array(self, price_list: list) -> np.ndarray:
        """Converts a list of prices to a NumPy float array"""
        return np.array(price_list, dtype=float)

    def calculate_total_cost(self, prices: np.ndarray, quantities:
np.ndarray) -> float:
        """Calculates the total cost of items using element-wise
multiplication"""
        return float(np.sum(prices * quantities))

    def apply_discount(self, cost_array: np.ndarray, discount_percent: float)
-> np.ndarray:
        """Applies discount to each item and rounds the result"""
        discounted = cost_array * (1 - discount_percent / 100)
        return np.round(discounted, 1)
    def filter_items_above_threshold(self, cost_array: np.ndarray, threshold:
float) -> np.ndarray:
        """Returns a filtered array of items above the given threshold"""
        return cost_array[cost_array > threshold]

# Driver Code
if __name__ == "__main__":
    sb = ShoppingBill()
    q = int(input())
    for _ in range(q):
        cmd = input().split()
        if cmd[0] == "price":
            price_list = list(map(float, cmd[1:]))
            print(sb.create_price_array(price_list))
        elif cmd[0] == "total":
            prices = np.array(list(map(float, input().split())))
            quantities = np.array(list(map(int, input().split())))
            print(sb.calculate_total_cost(prices, quantities))
        elif cmd[0] == "discount":
            cost_array = np.array(list(map(float, input().split())))
            discount = float(input())
            print(sb.apply_discount(cost_array, discount))
        elif cmd[0] == "filter":
            arr = np.array(list(map(float, input().split())))
            threshold = float(input())
            print(sb.filter_items_above_threshold(arr, threshold))
        else:
            print("Invalid command.")
```

□ Test Cases and Marks Allocation

Test Case ID	Test Case Description	Associated Function(s)	Marks
TC1	Creating a price array from list	create_price_array()	□ 2 Marks
TC2	Calculating total cost using price × quantity	calculate_total_cost()	□ 2 Marks
TC3	Applying discount percentage correctly	apply_discount()	□ 2 Marks
HTC1	Edge case: 100% discount reduces values to 0	apply_discount()	□ 2 Marks
HTC2	Filtering values above threshold (boundary value included)	filter_items_above_threshold()	□ 2 Marks
□ TOTAL	All test cases passed	–	□ 10 Marks

□ Visible Test Cases (3)

□ Test Case 1: Create Price Array
Input:
```
1
price 100 200 300
```
Output:
```
[100. 200. 300.]
```

□ Test Case 2: Calculate Total Cost
Input:
```
1
total
100 200
2 1
```
Output:
```
400.0
```

□ Test Case 3: Apply Discount
Input:
```
1
discount
100 200
10
```
Output:
```
[ 90. 180.]
```

□ Hidden Test Cases (2)

□ HTC1: Edge Discount 100%
Input:
```
1
discount
99
100
```
Output:
```
[0.]
```

□ HTC2: Filter with Edge Threshold
Input:
```
1
filter
30 50 100
50
```
Output:
```
[100.]
```