**SHLOK ANAND**

**23070521142**

**Final Task Practical 5 Cart Total Calculator**

**Problem Statement**

You are building a **shopping cart calculator**. Each cart item is represented as an **object** with: { id, name, price, qty, category }

The program must calculate:

1. **Subtotal** (sum of all items price × qty).

2. **Discounts**:

If quantity ≥ 3, apply **5% discount** on that item.

If category = "stationery" and subtotal of stationery items > 200, apply **10% category discount**.

3. **Final Total** after discounts.

**Steps**

1. **Create the cart**: An array of objects (3–5 products, different categories). 2. **Use map()** to calculate each item’s total (price × qty).

3. **Apply item-level discount** using map() or inside reduce().

4. **Use filter()** to extract stationery items and calculate their subtotal. 5. **Apply category-level discount** if applicable.

6. **Use reduce()** to calculate overall subtotal and final total.

7. **Use forEach()** to print a formatted receipt.

**Sample Input (Cart Example)**

[

{ id: 1, name: "Pen", price: 20, qty: 2, category: "stationery" },

{ id: 2, name: "Mug", price: 150, qty: 1, category: "kitchen" },

{ id: 3, name: "Notebook", price: 80, qty: 3, category: "stationery" }

]

**Expected Output (Approximate Format)**

Item: Pen (x2) = 40

Item: Mug (x1) = 150

Item: Notebook (x3) = 240 → discount applied

---------------------------------

Subtotal: 430

Item Discounts: 12

Stationery Discount: 26.8

Final Total: 391.2

**Reminders**

● Don’t forget **quantity × price** when calculating totals.

● Discounts must be **subtracted after subtotal** is found.

● Use reduce() effectively to **accumulate sums**.

● Break the problem into **small reusable functions**.

● Format money properly (e.g., two decimal places).

**Note:** This is an **implementation lab**. Follow the hints carefully and **write the full program yourself**.

**Detailed Steps:**

**Part A – Setup the Cart**

1. Create an array named cart.

Each element should be an object with properties:

{ id, name, price, qty, category }.

Example categories: stationery, kitchen, electronics.

Write your cart array below:

let cart = [

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

];

Checkpoint: At least **3–4 items** in your cart.

**Part B – Subtotal per Item**

1. Use map() to calculate the **subtotal** for each item (price × qty). 2. Print each item’s subtotal.

Formula hint: item.price \* item.qty

let itemTotals = cart.map(item => \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ );

console.log(itemTotals);

Checkpoint: Does your array show the correct subtotal values?

**Part C – Item-Level Discount**

Rule: If qty ≥ 3, apply a **5% discount** on that item.

Hint: Use a **conditional (if / ternary)** inside your calculation.

Fill in logic:

let discountedItemTotals = cart.map(item => {

let subtotal = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ) {

subtotal = subtotal - (subtotal \* 0.05);

}

return subtotal;

});

Checkpoint: Test with an item having qty = 3 or more.

**Part D – Category-Level Discount**

Rule: If total **stationery subtotal > 200**, apply **10% off stationery subtotal**. 1. Use filter() to extract stationery items.

2. Use reduce() to sum their total.

3. If the total > 200, calculate discount.

Fill-in:

let stationeryItems = cart.\_\_\_\_\_\_\_\_\_\_(item => item.category === "stationery"); let stationeryTotal = stationeryItems.\_\_\_\_\_\_\_\_\_\_( (sum, item) => sum + (item.price \* item.qty), 0);

if( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ) {

let stationeryDiscount = stationeryTotal \* 0.10;

console.log("Stationery Discount:", stationeryDiscount);

}

Checkpoint: Try with a stationery-heavy cart.

**Part E – Final Total**

1. Use reduce() to get the **cart subtotal**.

2. Subtract **item-level discounts**.

3. Subtract **category-level discount**.

4. Print the final total with two decimal places.

Reminder: Number(value.toFixed(2)) can help format decimals.

**Sample Input (For Testing)**

[

{ id: 1, name: "Pen", price: 20, qty: 2, category: "stationery" },

{ id: 2, name: "Mug", price: 150, qty: 1, category: "kitchen" },

{ id: 3, name: "Notebook", price: 80, qty: 3, category: "stationery" }

]

**Expected Output (Format Example)**

Item: Pen (x2) = 40

Item: Mug (x1) = 150

Item: Notebook (x3) = 240 → discount applied

---------------------------------

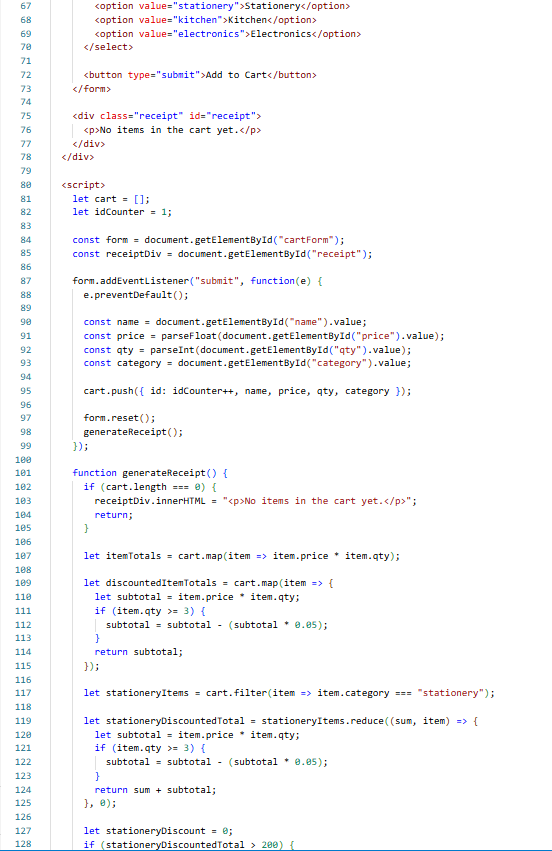
Subtotal: 430

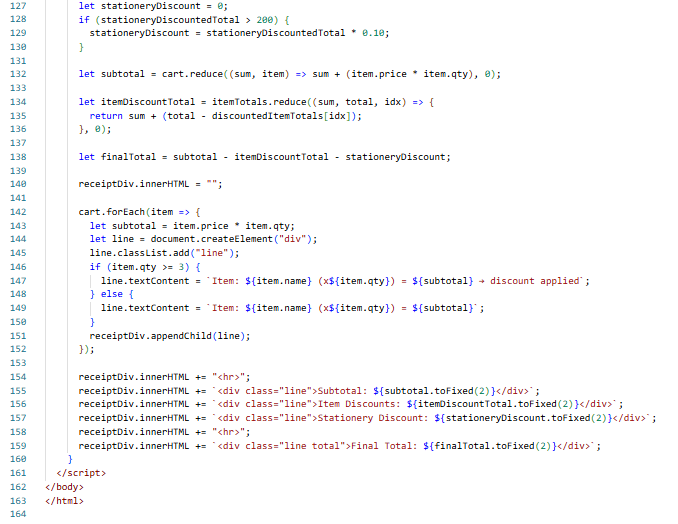
Item Discounts: 12

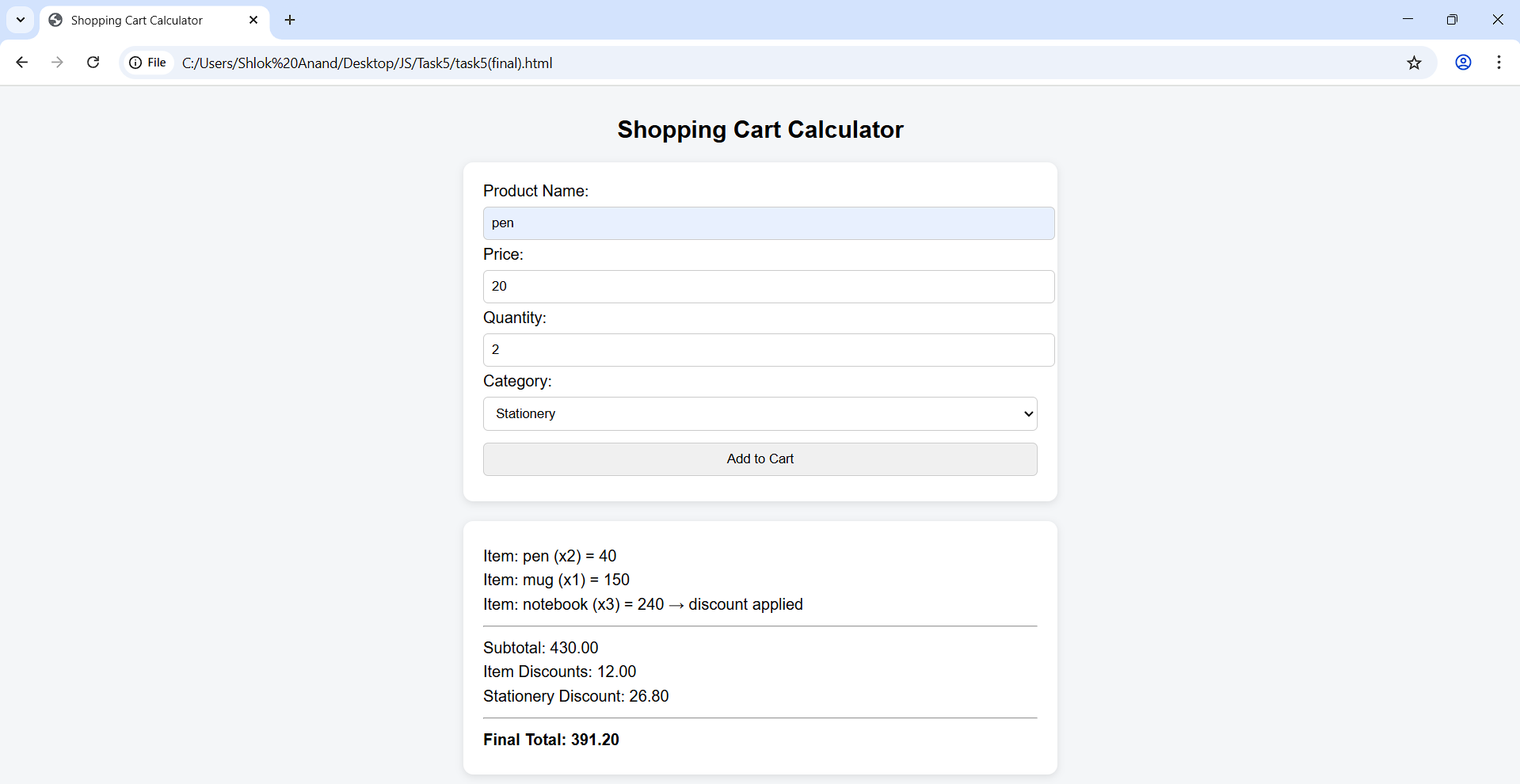
Stationery Discount: 26.8

Final Total: 391.2

****

****

****

****

**Solve the following**

● Which function (map, filter, reduce, forEach) was **hardest** to apply? Why?

Ans) The most difficult operation to implement was reduce() as it involves using an accumulator and making very careful choices about how the values can be added at each step. In contrast to map or filter that simply result in transformed arrays, reduce is more abstract and was difficult to calculate subtotals and discounts simultaneously. It required additional consideration to establish the appropriate initial value and to be sure the logic was right.

● How could this program be extended (e.g., tax, coupons, shipping)?

Ans) The program may be expanded by including the calculation of taxes (e.g. the percentage of GST/VAT applied to the total final cost), promo codes or coupons which would provide fixed or percentage based discounts, and shipping fees based on the order size or place of delivery. The cart might also be made more real and more like real-world e-commerce systems with added features such as loyalty points, bulk discounts, or free shipping thresholds.

**Important:** This worksheet contains hints only. You must **write the complete working program** yourself.