1357. Apply Discount Every n Orders

Description

There is a supermarket that is frequented by many customers. The products sold at the supermarket are represented as two parallel integer arrays `products` and `prices`, where the `i th` product has an ID of `products[i]` and a price of `prices[i]`.

When a customer is paying, their bill is represented as two parallel integer arrays `product` and `amount`, where the `j th` product they purchased has an ID of `product[j]`, and `amount[j]` is how much of the product they bought. Their subtotal is calculated as the sum of each `amount[j] * (price of the j th product)`.

The supermarket decided to have a sale. Every `n th` customer paying for their groceries will be given a **percentage discount**. The discount amount is given by `discount`, where they will be given `discount` percent off their subtotal. More formally, if their subtotal is `bill`, then they would actually pay `bill * ((100 - discount) / 100)`.

Implement the `Cashier` class:

- `Cashier(int n, int discount, int[] products, int[] prices)` Initializes the object with `n`, the `discount`, and the `products` and their `prices`.
- `double getBill(int[] product, int[] amount)` Returns the final total of the bill with the discount applied (if any). Answers within `10 -5` of the actual value will be accepted.

Example 1:

```
Input
["Cashier","getBill","getBill","getBill","getBill","getBill","getBill","getBill"]
[[3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]],[[1,2],[1,2]],[[3,7],[10,10]],[[1,2,3,4,5,6,7],[1,1,1,1,1,1,1]],[[4],[10]],[[7,3],[10,10]],
[[7,5,3,1,6,4,2],[10,10,10,9,9,9,7]],[[2,3,5],[5,3,2]]]
Output
[null,500.0,4000.0,800.0,4000.0,4000.0,7350.0,2500.0]
Explanation
Cashier cashier = new Cashier(3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]);
cashier.getBill([1,2],[1,2]);                        // return 500.0. 1 st customer, no discount.
                                                     // bill = 1 * 100 + 2 * 200 = 500.
cashier.getBill([3,7],[10,10]);                      // return 4000.0. 2 nd customer, no discount.
                                                     // bill = 10 * 300 + 10 * 100 = 4000.
cashier.getBill([1,2,3,4,5,6,7],[1,1,1,1,1,1,1]);    // return 800.0. 3 rd customer, 50% discount.
                                                     // Original bill = 1600
                                                     // Actual bill = 1600 * ((100 - 50) / 100) = 800.
cashier.getBill([4],[10]);                           // return 4000.0. 4 th customer, no discount.
cashier.getBill([7,3],[10,10]);                      // return 4000.0. 5 th customer, no discount.
cashier.getBill([7,5,3,1,6,4,2],[10,10,10,9,9,9,7]); // return 7350.0. 6 th customer, 50% discount.
                                                     // Original bill = 14700, but with
                                                     // Actual bill = 14700 * ((100 - 50) / 100) = 7350.
cashier.getBill([2,3,5],[5,3,2]);                    // return 2500.0.  7 th customer, no discount.
```

Constraints:

- `1 <= n <= 10 4`
- `0 <= discount <= 100`
- `1 <= products.length <= 200`
- `prices.length == products.length`
- `1 <= products[i] <= 200`
- `1 <= prices[i] <= 1000`
- The elements in `products` are **unique**.
- `1 <= product.length <= products.length`
- `amount.length == product.length`
- `product[j]` exists in `products`.
- `1 <= amount[j] <= 1000`
- The elements of `product` are **unique**.
- At most `1000` calls will be made to `getBill`.
- Answers within `10 -5` of the actual value will be accepted.