2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

ExactChange PROBLEM STATEMENT

A store has a special policy on what to do when a cashier does not have the exact change to give to a customer making a purchase. According to this policy, the cashier can reduce the purchase price to the highest amount for which he has the exact change. The customer is given an automatic discount in the amount of that reduction. Consider an example: a customer is buying a $17.99 item, and hands the cashier a $20 bill. The cashier has a roll of quarters (twenty $0.25 coins), so he cannot give the customer the exact change of $2.01. The cashier must reduce the price to $17.75, and gives the customer $2.25 back. Therefore, the customer gets a $0.24 discount. The policy limits the value of the automatic discount to $10.00: in cases when a transaction would result in an automatic discount of $10.00 or more, the cashier goes to the bank and gets the change.

Given the price of a product, a list of bills and coins the customer has, and a list of bills and coins the cashier has, determine the value of the automatic discount (if any) that the customer gets purchasing the product. Assume that the customer is cooperating with the cashier (i.e. he is not hiding bills or coins he has in order to maximize his automatic discount).

DEFINITION
Class: ExactChange
Method: discount
Parameters: int, String[], String[]
Returns: int
Method signature (be sure your method is public): int discount(int price, String[] customersMoney, String[] cashiersMoney);

price is the price of the product expressed in cents.
Both customersMoney and cashiersMoney consist of elements formatted as "count:denomination" (here and below quotes are for clarity only). count is an integer representing the number of bills or coins of a particular denomination the customer or the cashier have; denomination is an integer representing the denomination of the bill or the coin, expressed in cents; there is exactly one colon ':' separating the count and denomination. For example, "12:2000" means 12 $20 bills; "5:25" means five quarters, etc.
Your method should return the amount of the automatic discount that the customer gets (if any). When it is possible to arrive at the exact change, or when the discount is greater than or equal to the $10.00 limit, your method should return zero.

TopCoder will ensure the validity of the inputs. Inputs are valid if all of the following criteria are met:
- price is in the range between 1 and 50,000 (1 penny to $500), inclusive.
- Customer is carrying a maximum of $500 (i.e. the total you get by adding up all count*denomination in customersMoney does not exceed 50,000).
- customersMoney contains between 0 and 50 elements, inclusive.
- cashiersMoney contains between 0 and 50 elements, inclusive.
- Elements of cashiersMoney and customersMoney are formatted as "count:denomination", where count and denomination are integers.
- Elements of cashiersMoney and customersMoney do not exceed 50 characters in length.
- For each element of cashiersMoney or customersMoney, count is in the range from 1 to 50, inclusive.

- For each element of cashiersMoney or customersMoney, denomination is in the range from 1 to 10,000, inclusive.
- cashiersMoney does not contain elements with duplicate denominations.
- customersMoney does not contain elements with duplicate denominations.

NOTES
- Note that there is no explicit upper limit on the total value of the cashiersMoney.
- When the customer does not have enough money to pay for the item, your method should return 0.
- Your method should not expect the denominations to be real (like, $1, $5, $10, etc.) For example, "10:99" is a valid entry of cashiersMoney or customersMoney.
- Your method should not expect cashiersMoney or customersMoney to be sorted.
- Both count and denomination may have leading zeros, which your method should ignore (see examples).

EXAMPLES

1. price= 13, customersMoney={"2:1","3:5"}, cashiersMoney= {}. The customer has more than 13 cents, and therefore can pay for the item. However, since the customer does not have the exact change, the highest amount he can pay without exceeding the price is 12 cents. Cashier has no change money, therefore the customer pays 12 cents for a resulting discount of 1 cent. Your method should return 1.

2. price= 999, customersMoney={"1:2000","1:2500"}, cashiersMoney= {"50:10000"}. The customer has enough money to pay for the item, but the value of his smallest bill is greater than the item's price. Since the cashier does not have anything smaller than $100 bills, the customer is getting the item for free. Your method should return 999.

3. price= 1000, customersMoney={"1:2000"}, cashiersMoney= {}. The customer can pay for the item, but does not have the exact change. The cashier cannot give the customer the item for free, because the resulting discount would be equal to $10.00, which is greater than the highest allowed discount. Your method should return 0.

4. price= 899, customersMoney={"1:2000"}, cashiersMoney= {"50:100", "50:25"}. Your method should return 24.

5. price= 30000, customersMoney={"40:900"}, cashiersMoney= {}. Your method should return 300.

6. price= 13, customersMoney={"02:01","02:05"}, cashiersMoney= {"1:1000","2:2000","3:10000"}. Your method should return 0, because the customer can't pay for the item.

7. Timeout test: price= 12345, customersMoney= cashiersMoney= {"50:1", "50:2", "50:3", "50:5", "50:7", "50:11", "50:13", "50:17", "50:19", "50:23", "50:29", "50:31", "50:37", "50:41", "50:43", "50:47", "50:53", "50:59", "50:61", "5:67", "5:71", "5:73", "5:79", "5:83", "5:89", "5:97", "5:101", "5:103", "5:107", "5:109", "5:113", "5:127", "5:131", "5:137", "5:139", "5:149", "5:151", "5:157", "5:163", "5:167", "5:173", "5:179", "5:181", "5:191", "5:193", "5:197", "5:199", "5:211", "5:223", "5:277"}. Your method should return 0.

```
Example 7 input formatted to allow copy/paste
50:1,50:2,50:3,50:5,50:7,50:11,50:13,50:17,50:19,50:23,50:29,50:31,50:37,50:41,50
:43,50:47,50:53,50:59,50:61,5:67,5:71,5:73,5:79,5:83,5:89,5:97,5:101,5:103,5:107,
5:109,5:113,5:127,5:131,5:137,5:139,5:149,5:151,5:157,5:163,5:167,5:173,5:179,5:1
81,5:191,5:193,5:197,5:199,5:211,5:223,5:277
```