YONSEI UNIVERSITY

CSI3108-01 2016. 10. 05

Programming HW#3

Max 20 points

Due on Oct. 11(Tue), 2016, by 5pm

The problem is to minimize the total number of coins that customer has to pay and receive, given that the cashier has an adequate supply of all coins.

Suppose that the set of coins comprises 5c, 10c, 20c, 50c, $1, and $2. If we (as customers) should to pay 55c and we do not hold a 50c coin, we could pay this as 2*20c + 10c + 5c to make a total of 4 coins. If we tender $1 instead, we will receive 45c in change which also involves 4 coins ($1 coin plus 2*20c + 5c). However, if we pay $1.05 ($1 + 5c), we get 50c change and the total number of coins that changes hands is only 3 ($1, 5c and 50c coins).

Write a java program that will read in the resources available to you and the amount of the purchase and will determine the minimum number of coins that change hands. Assume that $1 and $2 are coins, 100 and 200 (Check examples below).

Input

The test cases consist of the following format. In the first line, the number of test cases is given. Starting from the 2nd line, each test case is provided in 3 lines. The first line has the number of coin types and the amount of money that customer has to pay (**at most 500c, $5.00**). The second line has the coin system, listing the denominations of the coins of the system in increasing sequence. The last line has the numbers of coins available for customers in the order given from line 2.

Assume that the total value of the coins will always be sufficient to make up the amount for each given coin system and the amount will always be achievable.

Output

For each test case, print out the **minimum number of coins** that change hands followed by the tendering info and the change info as shown in the sample output.

Sample input

```
20
6 95                        // six types of coin, have to pay 95c
5 10 20 50 100 200   // 5c, 10c, 20c, 50c, $1, and $2
2 4 2 2 1 0         // two 5c coins, four 10c coins, ..., zero 200c coin
6 55
5 10 20 50 100 200
2 4 2 0 1 0
...
```

Sample output

```
2 0 0 0 0 1 0 1 0 0 0 0 0   // 2 / 0 0 0 0 1 0 / 1 0 0 0 0 0 (means number of coin that customer payed and received are "2" / payed 100c / received 5c)
3 1 0 0 0 1 0 0 0 0 1 0 0   // 3 / 1 0 0 0 1 0 / 0 0 0 1 0 0
...
```