166 Making Change

Given an amount of money and unlimited (almost) numbers of coins we know that an amount of money may be made up in a variety of ways. A more interesting problem arises when goods are bought and need to be paid for, with the possibility that change may need to be given. Given the finite resources of most wallets nowadays, we are constrained in the number of ways in which we can make up an amount to pay for our purchases—assuming that we can make up the amount in the first place, but that is another story.

The problem we will be concerned with will be to minimise the number of coins that change hands at such a transaction, given that the shopkeeper has an adequate supply of all coins. (The set of New Zealand coins comprises 5c, 10c, 20c, 50c, \$1 and \$2.) Thus if we need 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 we will receive 45c in change which also involves 4 coins, but if we tender \$1.05 (\$1 + 5c), we get 50c change and the total number of coins that changes hands is only 3.

Write a 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.

Input

Input will consist of a series of lines, each line defining a different situation. Each line will consist of 6 integers representing the numbers of coins available to you in the order given above, followed by a real number representing the value of the transaction, which will always be less than \$5.00. The file will be terminated by six zeroes (0 0 0 0 0 0). The total value of the coins will always be sufficient to make up the amount and the amount will always be achievable, that is it will always be a multiple of 5c.

Output

Output will consist of a series of lines, one for each situation defined in the input. Each line will consist of the minimum number of coins that change hands right justified in a field 3 characters wide.

Sample input

```
2 4 2 2 1 0  0.95
2 4 2 0 1 0  0.55
0 0 0 0 0 0
```

Sample output

```
  2
  3
```