Problem A : Bicycle

(blue balloon)

Most bicycle speedometers work by using a Hall Effect sensor fastened to the front fork of the bicycle. A magnet is attached to one of the spokes on the front wheel so that it will line up with the Hall Effect switch once per revolution of the wheel. The speedometer monitors the sensor to count wheel revolutions. If the diameter of the wheel is known, the distance traveled can be easily be calculated if you know how many revolutions the wheel has made. In addition, if the time it takes to complete the revolutions is known, the average speed can also be calculated. For this problem, you will write a program to determine the total distance traveled (in miles) and the average speed (in Miles Per Hour) given the wheel diameter, the number of revolutions and the total time of the trip. You can assume that the front wheel never leaves the ground, and there is no slipping or skidding.

Standard Input

Input consists of multiple datasets, one per line, of the form:

diameter revolutions time

The *diameter* is expressed in inches as a floating point value. The *revolutions* is an integer value. The *time* is expressed in seconds as a floating point value. Input ends when the value of *revolutions* is *0* (zero).

Standard Output

For each data set, print:

Trip #N: distance MPH

Of course *N* should be replaced by the data set number, *distance* by the total distance in miles (accurate to 2 decimal places) and *MPH* by the speed in miles per hour (accurate to 2 decimal places). Your program should not generate any output for the ending case when *revolutions* is 0.

Constants

For p use the value: 3.1415927.

There are 5280 feet in a mile.

There are 12 inches in a foot.

There are 60 minutes in an hour.

There are 60 seconds in a minute.

There are 201.168 meters in a furlong.

Sample Input	Sample Output
26 1000 5 27.25 873234 3000 26 0 1000	Trip #1: 1.29 928.20 Trip #2: 1179.86 1415.84

Problem B: Binary numbers

(red balloon)

Adding binary numbers is a very simple task, and very similar to the longhand addition of decimal numbers. As with decimal numbers, you start by adding the bits (digits) one column at a time, from right to left. Unlike decimal addition, there is little to memorize in the way of rules for the addition of binary bits:

```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 10
1 + 1 + 1 = 11
```

Just as with decimal addition, when the sum in one column is a two-bit (two-digit) number, the least significant figure is written as part of the total sum and the most significant figure is "carried" to the next left column. Consider the following examples:

```
                  11  1  <-- Carry bits --> 1   11
  1001101           1001001                  1000111
+ 0010010         + 0011001                + 1010110
---------         ---------                ---------
  1011111           1100010                 10011101
```

The addition problem on the left did not require any bits to be carried, since the sum of bits in each column was either **1** or **0**, not 10 or 11. In the other two problems, there definitely were bits to be carried, but the process of addition is still quite simple.

Standard Input

The first line of input contains an integer N, (**1 <= N <= 1000**), which is the number of binary addition problems that follow. Each problem appears on a single line containing two binary values separated by a single space character. The maximum length of each binary value is 80 bits (binary digits). Note: The maximum length result could be 81 bits (binary digits).

Standard Output

For each binary addition problem, print the problem number, a space, and the binary result of the addition. Extra leading zeroes must be omitted.

Sample Input	Sample Output
3 1001101 10010 1001001 11001 1000111 1010110	1 1011111 2 1100010 3 10011101

Problem C : Change
(yellow balloon)

J.P. Flathead's Grocery Store hires cheap labor to man the checkout stations. The people he hires (usually high school kids) often make mistakes making change for the customers. Flathead, who's a bit of a tightwad, figures he loses more money from these mistakes than he makes; that is, the employees tend to give more change to the customers than they should get.

Flathead wants you to write a program that calculates the number of quarters ($0.25), dimes ($0.10), nickels ($0.05) and pennies ($0.01) that the customer should get back. Flathead always wants to give the customer's change in coins if the amount due back is $5.00 or under. He also wants to give the customers back the smallest total number of coins. For example, if the change due back is $1.24, the customer should receive 4 quarters, 2 dimes, 0 nickels, and 4 pennies.

Standard Input

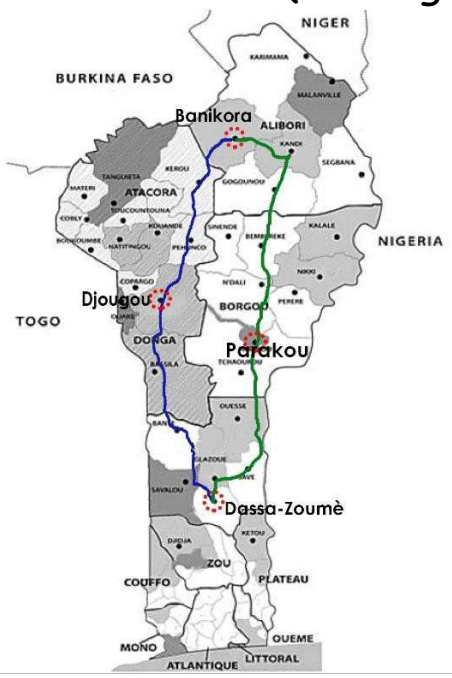
The first line of input contains an integer **N** which is the number of datasets that follow. Each dataset consists of a single line containing a single integer which is the change due in cents, **C**, ($1 \leq C \leq 500$).

Standard Output

For each dataset, print out the dataset number, a space, and the string: Q QUARTER(S), D DIME(S), n NICKEL(S), P PENNY(S) Where Q is he number of quarters, D is the number of dimes, n is the number of nickels and P is the number of pennies.

Sample Input	Sample Output
3 124 25 194	1 4 QUARTER(S), 2 DIME(S), 0 NICKEL(S), 4 PENNY(S) 2 1 QUARTER(S), 0 DIME(S), 0 NICKEL(S), 0 PENNY(S) 3 7 QUARTER(S), 1 DIME(S), 1 NICKEL(S), 4 PENNY(S)

Problem D : The fastest road to banikoara (Orange balloon)



Codjo: I am ready for the trip to Banikoara

Bossi: Let us go through Parakou

Assiba: No, the fastest road to go to Banikoara is through Djougou

You are responsible to write a program which, given a list of towns and distances separating these towns, gives the shortest distance to travel from a town A to a town B.

Standard Input

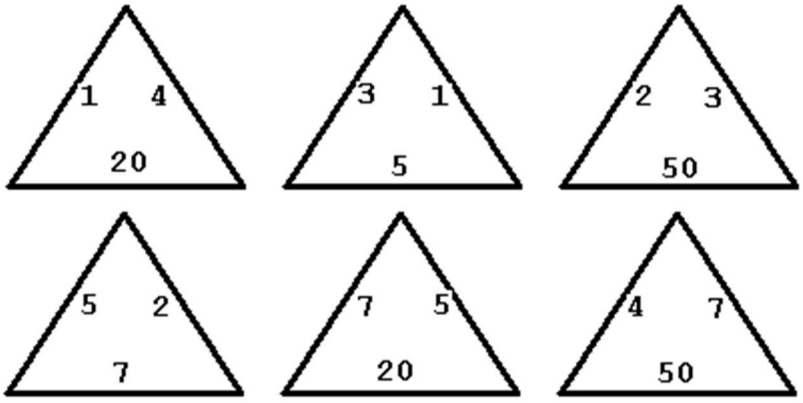
The first line of input contains a single integer **P**, ($1 \leq \mathbf{P} \leq 1000$), which is the number of data sets that follow. Each data set begins with a line containing the number **N** of the remaining lines in the dataset ($1 \leq \mathbf{N} \leq 500$), followed by a space, followed by the name of a departure town, followed by a space, followed by the name of a destination town. Each of these N lines contains a name of a departure town, followed by a space, followed by a name of a destination town, followed by the distance (in kilometers) between the two towns. The distance will be an integer and the name of town will be a string formed with characters [a-z] [A-Z] and with the sign"-".

Standard Output

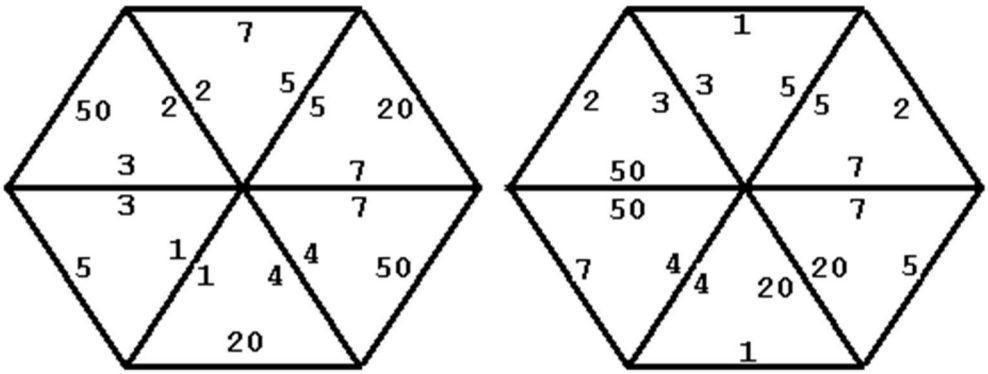
For each data set, you must generate a single output line containing the name of a town A and a space, followed by the name of town B, followed by a space, followed by the shortest distance to travel between towns A and B.

Sample Input	Sample Output
1 4 Dassa-Zoume Banikoara Dassa-Zoume Djougou 270 Banikoara Djougou 211 Parakou Banikoara 284 Parakou Dassa-Zoume 225	Dassa-Zoume Banikoara 481

Problem E : The Triangle Game
(Purple balloon)



In the triangle game you start off with six triangles numbered on each edge, as in the example above. You can slide and rotate the triangles so they form a hexagon, but the hexagon is only legal if edges common to two triangles have the same number on them. You may not flip any triangle over. Two legal hexagons formed from the six triangles are illustrated below.



The score for a legal hexagon is the sum of the numbers on the outside six edges.

Your problem is to find the highest score that can be achieved with any six particular triangles.

Standard Input

The input file contain one or more data sets. Each data set is a sequence of six lines with three integers from 1 to 100 separated by blanks on each line. Each line contains the numbers on the triangles in clockwise order. Data sets are separated by a line containing only an asterisk. The last data set is followed by a line containing only a dollar sign.

Standard Output

For each input data set, the output is a line containing only the word "none" if there are no legal hexagons or the highest score if there is a legal hexagon.

Sample Input	Sample Output
1 4 20 3 1 5 50 2 3 5 2 7 7 5 20 4 7 50 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 60 6 10 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 10 6 60 $	152 21 none