ACM-UCLA Local Programming Contest 2012

October 13, 2012

Universidad Centroccidental Lisandro Alvarado

This problem-set contains 8 problems; pages are numbered from 1 to 12.

Problem-Setter: Camilo Romero

Problem A

A chips game

Filename: `chips.c`, `chips.cpp` *or* `chips.java`
You must read from standard input and print to standard output.

A chips game is a simple game typically played by two players. One version of the game calls for each player to choose a unique three-coin sequence such as **HEADS TAILS HEADS (HTH).** A fair coin is tossed sequentially some number of times until one of the two sequences appears. The player who chose the first sequence to appear wins the game.

For this problem, you will write a program that implements a variation on the *chips game*. You will read a sequence of 40 coin tosses and determine how many times each *three-coin* sequence appears. Obviously there are eight such three-coin sequences: **TTT, TTH, THT, THH, HTT, HTH, HHT** and **HHH.** Sequences may overlap. For example, if all 40 coin tosses are heads, then the sequence **HHH** appears 38 times.

Input

The first line of input contains a single integer C ($1 \leq C \leq 1000$), which is the number of data sets that follow. Each data set consists of 2 lines. The first line contains the data set number N. The second line contains the sequence of 40 coin tosses. Each toss is represented as an upper case **H** or an upper case **T**, for heads or tails, respectively. There will be no spaces on any input line.

Output

For each data set there is one line of output. It contains the data set number followed by a single space, followed by the number of occurrences of each three-coin sequence, in the order shown above, with a space between each one. There should be a total of 9 space separated decimal integers on each output line.

Input sample

```
4
1
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
2
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
3
HHTTTHHTTTHTHHTHHTTHTTTHHHTHTTHTTHTTTHTH
4
HTHTHHHTHHHTHTHHHHTTTHTTTTTHHTTTTHTHHHHT
```

Output sample

```
1 0 0 0 0 0 0 0 38
2 38 0 0 0 0 0 0 0
3 4 7 6 4 7 4 5 1
4 6 3 4 5 3 6 5 6
```

Problem B

NIM-B Sum

Filename: base`.c`*, base.*`cpp` *or base.*`java`
You must read from standard input and print to standard output.

The game of *NIM* is played with any number of piles of objects with any number of objects in each pile. At each turn, a player takes one or more (up to all) objects from one pile. In the normal form of the game, the player who takes the last object is the winner. There is a well-known strategy for this game based on the sum of NIM with base 2.

The NIM-B sum (NIM sum base **B**) of two non-negative integers **X** and **Y** (written ***f (B, X, Y)***) is computed as follows:

1. Write each of **X** and **Y** in base **B**.
2. Each digit in base **B** of the Nim-B sum is the sum modulo **B** of the corresponding digits in the base **B** representation of **X** and **Y.**

For example:

$f(2, 123, 456) = 1111011 \text{ ¤ } 111001000 = 110110011 = 435$

$f(3, 123, 456) = 11120 \text{ ¤ } 121220 = 102010 = 300$

$f(4, 123, 456) = 1323 \text{ ¤ } 13020 = 10303 = 307$

The strategy for normal form Nim is to compute the Nim-2 sum **T** of the sizes of all piles. If at any time, you end your turn with **T = 0**, you are guaranteed a **WIN**. Any opponent move must leave **T** not 0 and there is always a move to get **T** back to 0. This is done by computing ***f (2, T, PS)*** for each pile; if this is less than the pile size ***(PS)***, compute the difference between the **PS** and the sum of NIM with base 2 and remove it from that pile as your next move.

Write a program to compute ***f (B, X, Y).***

Input

The first line of input contains a single integer T ($1 \le T \le 1000$), which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by three spaces separated decimal integers, **B, X** and **Y**. ($2 \le B \le 2000000, 0 \le X \le 2000000, 0 \le Y \le 2000000$).

Output

For each data set there is one line of output. It contains the data set number followed by a single space, followed by *N*, the decimal representation of the Nim sum in base **B** of **X** and **Y**.

Input sample

```
4
1 2 123 456
2 3 123 456
3 4 123 456
4 5 123 456
```

Output sample

```
1 435
2 300
3 307
4 429
```

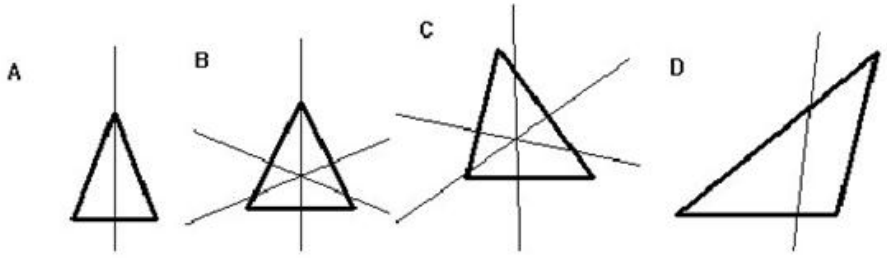
Problem C

Cake

Filename: `cake.c`, `cake.cpp` *or* `cake.java`
You must read from standard input and print to standard output.

For each birthday of twins Paola and Lilibeth, their mother bakes a special triangular birthday cake with a different shape each year. The sides of the cake are frosted with chocolate icing and the top is frosted with special pink icing. Paola and Lilibeth both like both types of icing and insist that they get their equal share of each.

Write a program to help Paola and Lilibeth cut the cake so they each get an equal share. A triangle equalizer is a line through a triangle that simultaneously divides the perimeter and area of the triangle into two equal pieces. If the triangle is isosceles, the bisector of the angle between two equal sides is such a line (A and B below). In general, a triangle may have 1, 2 or 3 such lines (2 requires special conditions not likely to be found in a random triangle).



Write a program to compute the equation of one such dividing line (it does not matter which one) for a non-degenerate input triangle. You should use double-precision calculations for intermediate values.

Input

The first line of input contains a single integer T ($1 \leq T \leq 1000$), which is the number of data sets that follow. Each data set consists of a single line containing a single decimal integer and 6 floating point numbers. The integer is the data set number (starting at 1). The floating-point numbers are the coordinates of the vertices of the input triangle: **x0, y0, x1, y1, x2, y2.**

Output

For each data set, there is a single line of output. The line contains a decimal integer giving the data set number followed by a single space, followed by 3 space-separated floating-

point numbers to 5 decimal places. The 3 floating-point values are the coefficients (A, B and C) of the equation of the equalizer line:

$$A * x + B * y = C$$

where $A*A + B*B = 1.0$ and $A >= 0$.

Input sample

```
4
1 0 0 4 10 8 0
2 0 0 10 8 8 0
3 0 0 -8 4.5 5 0
4 0 0 -5 4.6 5 0
```

Output sample

```
1 1.00000 0.00000 4.00000
2 0.99347 -0.11408 5.98771
3 0.45018 0.89294 0.81417
4 0.66369 0.74801 1.01564
```

Problem D

Geometric sums

Filename: geometric.c, geometric.cpp or geometric.java
You must read from standard input and print to standard output.

Into the geometric sums exists, the **n**th Triangular number, **T(n) = 1 + ... + n**, is the sum of the first **n** integers. It is the number of points in a triangular array with **n** points on side. For example T(4):

X
XX
XXX
XXXX

Write a program to compute the weighted sum of triangular numbers:

$$f(n) = SUM[k = 1..n; k * T(k + 1)]$$

Input

The first line of input contains a single integer N ($1 \leq N \leq 1000$), which is the number of data sets that follow. Each dataset consists of a single line of input containing a single integer $\boldsymbol{n}$ ($1 \leq n \leq 300$), which is the number of points on a side of the triangle.

Output

For each dataset, output on a single line the dataset number, (**1** through **N**), a blank, the value of **n** for the dataset, a blank, and the weighted sum, **W(n)**, of triangular numbers for **n**.

Input sample

```
4
3
4
5
10
```

Output sample

```
1 3 45
2 4 105
3 5 210
4 10 2145
```

Problem E

CryptoCR

Filename: cryptocr`.c`*, cryptocr*`.cpp` *or cryptocr*`.java`
You must read from standard input and print to standard output.

CryptoCR is an algorithm developed by a hacker called CR that permit encode message where one letter is simply replaced by another throughout the message. For example:

Encoded: **HPC PJVYMIY**

Decoded: **ACM CONTEST**

In the example above, **H=A, P=C, C=M, J=O, V=N, Y=T, M=E** and **I=S**. For this problem, you will decode messages.

Input

The first line of input contains a single integer N ($1 \leq N \leq 1000$), which is the number of data sets that follow. Each data set consists of two lines of input. The first line is the encoded message. The second line is a 26 character string of upper case letters giving the character mapping for each letter of the alphabet: the first character gives the mapping for **A**, the second for **B** and so on. Only upper case letters will be used. Spaces may appear in the encoded message, and should be preserved in the output string.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and the decoded message.

Input sample

```
2
HPC PJVYMIY
BLMRGJIASOPZEFDCKWYHUNXQTV
FDY GAI BG UKMY
KIMHOTSQYRLCUZPAGWJNBVDXEF
```

Output sample

```
1 ACM CONTEST
2 THE SKY IS BLUE
```

Problem F

The compass bank

Filename: `compass.c`, `compass.cpp` *or* `compass.java`
You must read from standard input and print to standard output.

Customers of the compass bank handle their banking transactions similar to the way they handle their taxes: be as terse as possible. As a result, when a customer writes a check or fills out a deposit or withdrawal form, they leave off the year on any date they write down. So, instead of writing: 09/20/2005, they would write: 9/20 and be done with it. In general, the year can be inferred since it will be relatively close to the date the transaction is actually processed by the bank.

Without going into the intricate details of how the compass bank calculates interest and banking fees (that is a problem for another time...), suffice to say the bank must determine the actual date the customer wrote on the check or form, and calculate the number of days prior (or in the future) the document is dated. You see, compass bankers, like their government officials, are overworked, so they may not get around to processing transactions for up to a week. The customers know this, so they often date their checks and forms a several days in the future –this complicates the bankers' duties as well.

Your job is to write a program to compare a date written on a check or form with the date the transaction is being processed, and, determine the full date the customer meant as well as how many days prior (or in the future) the document is dated.

Input

The first line of input contains an integer ***N*** which is the number of datasets that follow ($1 \leq N \leq 1000$). Each dataset consists of a single line containing two dates: the transaction date and the document date; there is a single space between them. The transaction date is of the form **M/D/Y** where **M** is the month number ($1 \leq M \leq 12$), **D** is the day of month ($1 \leq D \leq md1$), and **Y** is the year ($2000 \leq Y \leq 2200$). The document date is of the form **m/d** where **m** is the month number ($1 \leq m \leq 12$), and **d** is the day of month ($1 \leq d \leq md2$). The values of ***md1*** and ***md2*** will not exceed the number of days in the respective months ***M*** and ***m.***

Output

For each dataset print out the dataset number followed by a space followed by the result of the date comparison as shown in the table below:

Result to print	Criteria
m/d/y IS *n* DAY(S) PRIOR	If the document date occurs before the transaction date and is within 7 days in the past.
m/d/y IS *n* DAY(S) AFTER	If the document date occurs after the transaction date and is within 7 days in the future.
SAME DAY	If the dates are the same
OUT OF RANGE	All other results not with +/- 7 days.

Notes: When printing the result date, **m/d/y**, you will have to determine the year value **y** ($1999 \leq Y \leq 2201$). This is not necessarily the same as the transaction date's year value **Y.** Since the compass taxation fiasco a couple of years back, the compass government decided to switch to the standard Gregorian calendar. As such, Gregorian leap year rules apply. A year is a leap year (February has 29 days instead of 28) if the year if evenly divisible by 4, except for century years (those ending in 00), which are leap years only if they are evenly divisible by 400. 2000 and 2004 are leap years, but 2100 and 2101 are not. For those who do not know, the months of January, March, May, July, August, October and December all have 31 days in them. February has 28 days (unless in a leap year, then it has 29). The remainder of the months has 30 days.

Input sample

```
7
11/20/2005 11/21
11/20/2005 11/17
11/20/2005 11/20
11/20/2005 11/13
11/20/2005 11/28
1/2/2005 12/30
12/31/2100 1/3
```

Output sample

```
1 11/21/2005 IS 1 DAY AFTER
2 11/17/2005 IS 3 DAYS PRIOR
3 SAME DAY
4 11/13/2005 IS 7 DAYS PRIOR
5 OUT OF RANGE
6 12/30/2004 IS 3 DAYS PRIOR
7 1/3/2101 IS 3 DAYS AFTER
```

Problem G

Next Permutation

Filename: `permutation.c`, `permutation.cpp` *or* `permutation.java`
You must read from standard input and print to standard output.

For this problem, you will write a program that takes a (possibly long) string of decimal digits, and outputs the permutation of those decimal digits that has the next *larger* value (as a decimal number) than the input number. For example:

123 → 132
279134399742 → 279134423799

It is possible that no permutation of the input digits has a larger value. For example: **987**.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$) which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by up to 80 decimal digits which is the input value.

Output

For each data set there is one line of output. If there is no larger permutation of the input digits, the output should be the data set number followed by a single space, followed by the string **BIGGEST**. If there is a solution, the output should be the data set number, a single space and the next larger permutation of the input digits.

Input sample

```
3
1 123
2 279134399742
3 987
```

Output sample

```
1 132
2 279134423799
3 BIGGEST
```

Problem H

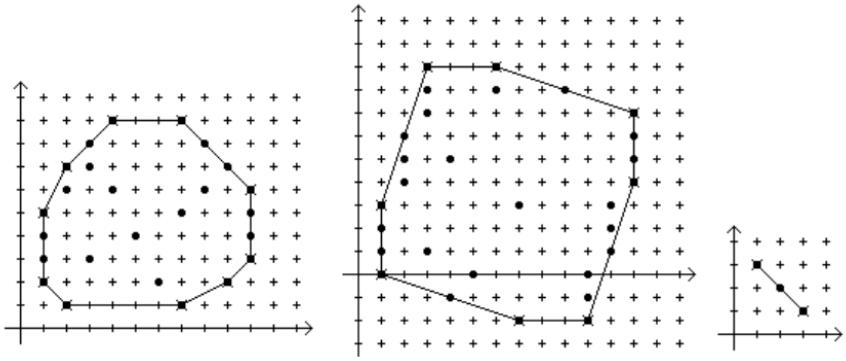
Convex Hull of Lattice Points

Filename: `hull.c`, `hull.cpp` *or* `hull.java`
You must read from standard input and print to standard output.

A ***lattice point*** is a point with ***integer*** coordinates. A ***lattice polygon*** is a polygon with all vertices lattice points.

A polygon is convex if any line segment between two points of the polygon is inside (or on the boundary of) the polygon. Equivalently, the interior angle at each polygon vertex is less than 180 degrees.

For a set **S**, of lattice points, the **convex hull** is the smallest convex (lattice) polygon which contains all points of the set. (The vertices of the convex hull must be members of the set of lattice points). If all the points are on a single straight line, the convex hull will be a line segment (a **degenerate** polygon – see rightmost diagram below). In the diagrams below, the points of the set are indicated by solid dots, the vertices of the convex hull by **X**'s and the convex hull is drawn connecting the vertices. Note that not all points on the convex hull polygon are vertices.



The vertices of a lattice polygon are in standard order if:

1. The first vertex is the one with the largest y value. If two vertices have the same y value, the one with the smaller x value is the first.
2. Vertices are given in clockwise order around the polygon.

Write a program that reads a set of lattice points and outputs the vertices of the convex hull of the points in standard order.

Input

The first line of input contains a single integer **P**, ($1 \leq P \leq 1000$) which is the number of data sets that follow. The first line of each data set contains the data set number, followed by a space, followed by a decimal integer giving the number of points **N**, ($3 \leq N \leq 50$), in the set. The remaining lines in the data set contain the points in the set; at most 5 points per line (the last line may have fewer). Each point consists of 2 spaces separated decimal integer values representing the x and y coordinates respectively.

Output

For each data set there are multiple lines of output. The first line contains a decimal integer giving the data set number followed by a single space, followed by a decimal integer giving the total number of vertices of the convex hull. The vertices of the convex hull follow, one per line in standard order. Each line contains the decimal integer x coordinate, a single space and the decimal integer y coordinate.

Input sample

```
4
1 25
2 1 7 1 1 2 9 2 1 3
10 3 1 4 10 4 1 5 10 5
2 6 10 6 2 7 9 7 3 8
8 8 4 9 7 9 6 2 3 3
5 4 7 5 8 6 4 6 3 7
2 30
3 9 6 9 3 8 9 8 3 7
12 7 2 6 12 6 2 5 12 5
2 4 12 4 1 3 11 3 1 2
11 2 1 1 11 1 1 0 10 0
4 -1 10 -1 7 -2 10 -2 5 0
7 3 4 5 6 8 3 1 2 6
3 3
3 1 2 2 1 3
4 6
1 3 19 1 4 2 2 1 11 2
10 1
```

Output sample

```
1 10
4 9
7 9
10 6
10 3
9 2
7 1
2 1
1 2
1 5
2 7
2 8
3 9
6 9
12 7
12 4
10 -2
7 -2
1 0
1 3
3 2
1 3
3 1
4 4
1 3
11 2
19 1
2 1
```