ACM-UCLA Programación Creativa 2012

Third Practice

June 16, 2012

Universidad Centroccidental Lisandro Alvarado

This problem-set contains 9 problems; pages are numbered from 1 to 10.

Problem A

Apple

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

It's about the fruit, seriously.

There is a stock of n apples, offered in a machine, for some reason the machine can only dispense 3, 6 or 11 apples in any time, the other options doesn't work. You were asked to see if it was possible to the machine to dispense all of its apples.

Input

You are given several cases, each one in a line with the value of n, ($1 \leq n \leq 10^6$). The last case is followed by $n = 0$, this case shouldn't be processed.

Output

For each case, print one line with "YES" if all the apples can be dispensed or "NO" otherwise (without quotes). See sample for the output format.

Input sample

```
5
8
9
14
0

```

Output sample

```
Case 1: NO
Case 2: NO
Case 3: YES
Case 4: YES

```

Problem B

Bits Problem III

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

Every integer can be represented in a binary system; we have that 5, in base 10, corresponds to 101 in base 2. Also, we can define the count of active bits to the amount of digits 1 that appears in the binary representation of a particular number, in this case, 5 would have two active bits.

The problem consists in, given two integers A and B, to find the number X between A and B, inclusive, with the maximum number of active bits. In case of a tie between two or more numbers, find the minimum of them.

Input

The input starts with an integer T ($T \leq 100$) which is the number of test cases. Then T lines follows, each one with a pair of integers A and B ($0 \leq A \leq B \leq 10^{18}$).

Output

For each case print one line with the value of X. See sample for the output format.

Input sample

```
5
0 10
3 4
5 6
0 0
1 100

```

Output sample

```
Case 1: 7
Case 2: 3
Case 3: 5
Case 4: 0
Case 5: 63

```

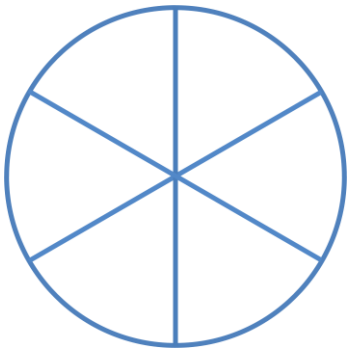
Problem C

Chiquipizzas

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

There is an issue in an Italian restaurant close to my home, they have a problem with the cutting machine, they want to cut their small pizzas in several pieces, so they will cut n slices through the diameter of the pizza, but they don't know how many pieces they are going to get.

For example, if they cut the pizza three times they get six pieces, as shown by the picture below.



Can you help them to predict the number of pieces they will get?

Input

There are several test cases in the input, each one with the value of n, an integer between 1 and 100, inclusive. The last case is followed by $n = 0$, this case shouldn't be processed.

Output

For each case you should print the number of pieces they are going to get. See sample for the output format.

Input sample

```
3
5
10
0

```

Output sample

```
Case 1: 6
Case 2: 10
Case 3: 20

```

Problem D

Dominoes

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

You are given a lot of dominoes. So you are asked to match pairs of dominoes, in fact you must calculate how different pairs you can make from the given dominoes.

Input

You are given several cases, the first line contains an integer T, the number of cases ($1 \leq T \leq 20$). So, for each case, the first line contains n ($1 \leq n \leq 28$), the number of dominoes, then n lines are followed, each one with two integers between 0 and 6, the values of each side of the domino.

Output

For each case, print the number of pair of dominoes that you can make from the given dominoes. See sample for the output format.

Input sample

```
1
4
1 2
4 3
6 5
5 4

```

Output sample

```
Case 1: 2

```

Problem E

Erotic Crocodiles from the Pink Lake

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

In case you don't know, that is an actual ICPC team name coming from the serious guys in Carabobo University.

There is a lake with pink water, and there are several naked crocodiles in it, of course they are usually naked, but for some reason someone tried to make emphasis on this. I just believe these people are frequently high or have severe mental issues, maybe both.

Anyway, there is a lake with crocodiles, and one of them is called Bob. Bob, the indecent crocodile, is looking to traverse the lake; he likes the leaves found in it, numbered from 1 to n, and wants to touch every leaf found in the water. To do so, he starts at leaf 1, and at every time he choose the closest non-visited leaf and start travelling to that leaf, in a straight line to it, and when he gets there he choose another one, and so on, until all the leaves are visited. In the case of a tie, he will choose the leaf with the lowest id number.

We ask you to tell us how much he will travel, given a particular configuration of the lake.

Input

The first line contains the integer T ($1 \leq T \leq 20$), the number of cases. T cases are followed, each one starts with an integer n in a line ($1 \leq n \leq 100$), the number of leaves, and then n more lines, with the position x, y of the i^{th} leaf, these are integers between 1 and 200. No two leaves will have the same position.

Output

Print for each case a floating-point number with two digits of precision, indicating how much Bob will travel. See sample for the output format.

Input sample

```
2
2
4 5
1 20
4
1 1
1 2
1 3
2 1

```

Output sample

```
Case 1: 15.30
Case 2: 4.24
```

Problem F

FPA

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

FPA is a game I just made up, and you can be glad that the rules are very simple! There are two players and goes through several turns, and at each turn, a magic ball says DA or DO. If it says DA then player one wins one point, otherwise player two gains that point, in both cases the other player keeps the previous score. If some of the player scores two points, the game is finished and he wins the game! However, if both players get 1 point each, then their scores are both decreased to zero.

It's mindless, useless and a lot of adjectives you surely can use to describe it. Still, it's a problem; you have p, the probability of player 1 winning a point in a turn. What is his probability of winning the game?

Input

There are several cases in the input. The first line contains T ($1 \leq T \leq 20$), the number of cases and then T lines are followed, each one describing a case with the value of p ($0 \leq p \leq 1$).

Output

For each case print a floating-point number with six digits of precision, indicating the probability of the player 1 of winning the game. See sample for the output format.

Input sample

```
5
0.3
0.4
0.5
0.631
1.0

```

Output sample

```
Case 1: 0.155172
Case 2: 0.307692
Case 3: 0.500000
Case 4: 0.745171
Case 5: 1.000000

```

Problem G

Guitar Tablatures

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

In case you have played guitar previously you may know about tablatures, still, if you don't have any idea about this, let me explain you.

But first, in western music, the chromatic scale is commonly used to give some identification to musical notes, it has main 12 key signatures, these are, sorted by their pitch,

```
C C# D D# E F F# G G# A A# B
```

They are then followed again by C, C# and so on, but in another octave, these reflect the fact that their frequencies are doubled compared to the previous one, but sound very similar.

In the case of a normal guitar, it has six strings; each one represents a note, E B G D A E, from the thinnest to the thickest string. Note that the first string is two octaves higher than the last one. This instrument also has (commonly) 22 frets, this allows given in which part of the string you press to change the sound of the note. So, if you press the first fret in the first string you get F, if you press the second fret in the same string you get F#, and so on. If you press the twelfth fret you get E again, but in a higher octave.

The next table shows the notes given the fret and string you press, as well the corresponding relative octave.

0	**1**	**2**	**3**	**4**	**5**
E 2	F 2	F# 2	G 2	G# 2	A 2
B 1	C 1	C# 1	D 1	D# 1	E 2
G 1	G# 1	A 1	A# 1	B 1	C 1
D 0	D# 0	E 1	F 1	F# 1	G 1
A 0	A# 0	B 0	C 0	C# 0	D 0
E 0	F 0	F# 0	G 0	G# 0	A 0

Now, a tablature allows to amateur guitarists who doesn't know how to read music sheets to learn a song by indicating the fret it must be pressed in a particular moment, for example, the example below shows some tablature,

```
E ------------------------------------------------------------------
B ---------------------5--------------------------------------------
G -----------5-------7---5-7-5-----5-----------------------------------
D ---5-6-7--------7------------7-7-----7----5-7-7---------2---2--------
A -7-------7-----------------------------------5-7---5-0---0--------
E ------------------------------------------------8----------------
```

As you see, at first you press the seventh fret in the fifth string, and then you took off the finger and press the fifth fret in the fourth string, and so on. In case of zero, you just play that string without pressing any fret. You may even press several frets in distinct strings at the same time.

The problem is, sometimes you want to change the key of the song to make it to sound higher or lower, you are given a base key and a new target key to convert the notes. You must print the updated tablature.

Input

The first line of input has T, the number of cases ($1 \leq T \leq 20$). Then for each case there are three values, n, the length of the tablature, S and R, the original and the target key, respectably. There may be several notes played at the same time, in case of these fret numbers have different length of digits then they will be right-aligned. Between notes played at different times there is at least one column of dashes.

Output

For each case print the case number, and then six lines with the updated guitar tablature. The changed notes must be at the same string of the original tablature, and the number of columns of dashes between the notes must be keep intact.

Input sample

```
1
29 C D
-----------------------------
-----------------------------
-5---------------------------
-----7----5-7-7---------2---2
----------------5-7---5-0---0
--------------------8--------

```

Output sample

```
Case 1:
-----------------------------
-----------------------------
-7----------------------------
-----9----7-9-9----------4---4
----------------7-9----7-2---2
---------------------10--------

```

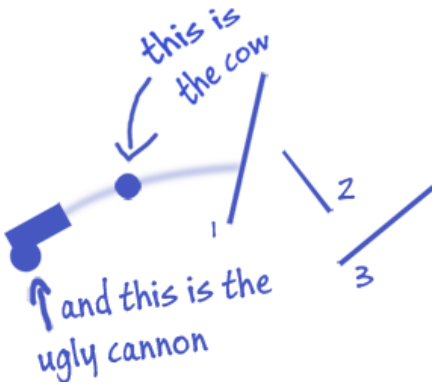
Problem H

High Flying Cows

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

This problem consists in study the art of throwing cows off by air. You see, there is one cannon at position (0,0) which can shoot cows in a field, with the aim of destroying some obstacles, all the system was built by a rich man whose "cows" gave him a very traumatic childhood. You don't want to know the details.

Well, the field consists of n obstacles, numbered from 1 to n; the i^{th} obstacle is defined by a line segment between (a_i, b_i) and (c_i, d_i), all in meters, and the cow is fired from the cannon, with v_y being the starting bottom-up velocity in the Y axis, and v_x a constant left-right velocity in the X axis, both given in m/s. The v_y velocity gradually decreases by 9.81 m/s every second and eventually turning the direction upside-down. For simplicity, let's suppose that the cow is a single point.



Your task is to find the first obstacle in which the cow shot will encounter, including extremes, or detect if it goes directly to the floor, i.e., doesn't intercept with any obstacle.

Input

The input consists of several cases, for each one, the first line contains three integers, n, v_x and v_y $(1 \leq n, v_x, v_y \leq 100)$, then it's followed by n lines, each describing an obstacle with four integers a_i, b_i, c_i and d_i $(1 \leq a_i, b_i, c_i, d_i \leq 100)$. The end of input is signalled by $n = v_x = v_y = 0$. The obstacles will not intercept for each case, not even at their extremes.

Output

For each case, print one line with the number of the first obstacle that the cow will encounter, or "floor" if it goes directly to the floor. See sample for the output format.

Input sample

```
3 10 40
10 20 30 40
50 60 70 80
90 30 40 60
0 0 0

```

Output sample

```
Case 1: 2

```

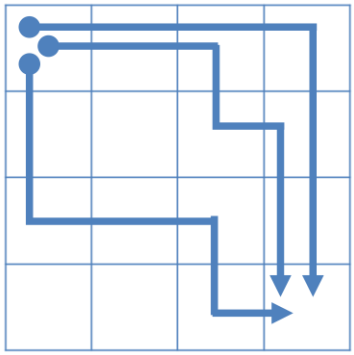
Problem I

Interview Question

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

I was asked to solve this problem in an interview; there were actually two versions, an easy one and another one slightly harder. This is the easy version.

Given a $N \times N$ square matrix, you are at the upper-left cell of the matrix, and you can go one cell at the right or go one down. Given those constraints, the problem consists in calculate how many ways you can get to the lower-right cell. The next figure shows some examples of how traverse a 4×4 matrix.



If you are curious, the other version was to solve the same problem but with the condition that some cells of the matrix are blocked, i.e., you can go through them.

Input

The input consists of several cases, each one in a line with an integer N $(1 \leq N \leq 20)$, the length of the matrix. The input end is signalled with $N = 0$, this case shouldn't be processed.

Output

For each case print one line with the solution. See sample for the output format.

Input sample

```
2
4
20
0

```

Output sample

```
Case 1: 2
Case 2: 20
Case 3: 35345263800

```