UTD PROGRAMMING CONTEST FOR HIGH SCHOOL STUDENTS

Welcome to UTD's Second Programming Contest for High School Students.

Here are the rules:

(1) Be nice to our lab and equipment, including the chairs and tables.
(2) Be nice to everyone in the lab and follow the directions of UTD student volunteers.
(3) Be quiet and don't celebrate loudly when you solve a problem.
(4) Don't bring food and drink into the lab. You can snack outside in the hallway.
(5) All submissions must be typed in by your team during the contest.
(6) Don't copy anything from a USB drive.
(7) Don't copy anything from the web.
(8) You can only use the web to access the contest scoreboard.
(9) Use one computer per team - do not switch computers.
(10) Answer the problems in any order.
(11) Use the Clarification System to ask questions about problems.
(12) Submit a solution as soon as you think it works.
(13) The team that solves the most problems in either category (Novice or Advanced) will win in that category.

Penalty points are used to break ties. For each problem solved, the number of penalty points is the sum of the submission time in minutes plus 20 points for each incorrect submission. So if you submit three incorrect solutions for problem 1 and finally submit a correct solution at 70 minutes, your penalty points for problem 1 will be 70 + 3*20 = 120.

A problem that you do not solve does not accrue any penalty points, no matter how many times you submit incorrect solutions.

(14) Have fun and work as a team.

Date: February 26th 2011.

Problem A: List of Conquests

Input file: Don.txt
Output: System.out
Time Limit: 5 seconds

In Act I, Leporello is telling Donna Elvira about his master's long list of conquests:

"This is the list of the beauties my master has loved, a list I've made out myself: take a look, read it with me. In Italy six hundred and forty, in Germany two hundred and thirty-one, a hundred in France, ninety-one in Turkey; but in Spain already a thousand and three! Among them are country girls, waiting-maids, city beauties; there are countesses, baronesses, marchionesses, princesses: women of every rank, of every size, of every age."
(Madamina, il catalogo questo)

Leporello's records are in chronological order, one entry per "beauty." He wants to create a new list of the number of "beauties" that Don Giovanni "loved" by country. He needs your help.

Input

The first line of the input contains a single integer $n \leq 2000$ giving the number of lines to follow. Each of the following lines contains no more than 75 characters, as follows: the first word is the name of a country. The rest of the line is made up of multiple words giving the name of a woman that Don Giovanni loved.

Output

The output consists of lines in alphabetical order. Each line starts with the name of a country, followed by a space, and then the total number of women Giovanni loved from that country.

Sample Input

```
3
Spain Donna Elvira
England Jane Doe Smith
Spain Donna Anna
```

Sample Output

```
England 1
Spain 2
```

Problem B: The Egg Seller

Input file: Eggs.txt
Output: System.out
Time Limit: 3 seconds

A puzzle from medieval times goes like this. A farmer stands in the market square selling chickens eggs from his basket. One day a curious thing happens. A series of customers buy eggs from him, leaving his basket empty (that's not the curious part.)

Each of these customers bought a number of eggs equal to half of the eggs in the basket at that time, plus half an egg. Without rounding, each customer bought a whole number of eggs.

If there were N customers, how many eggs did the farmer begin with?

Write a program to solve this puzzle.

Input:
A single integer, M, on a line by itself gives the number of test cases to follow. Then M lines follow, each containing a single integer N, ($N < 20$), giving the number of customers in this test case.

Output:
For each test case, output a single line giving the initial number of eggs in the farmers basket.

Sample Input:

```
1
2
```

Sample Output:

```
3
```

Problem C: The Train Swapper

Input file: Train.txt
Output: System.out
Time Limit: 5 seconds

At an old railway station, you may still encounter one of the last remaining "train swappers". A train swapper is an employee of the railroad, whose job is to rearrange the railcars of trains.

Once the railcars are arranged in the optimal order, all the train driver has to do, is drop the railcars off, one by one, at the stations for which the load is meant.

The title "train swapper" stems from the first person who performed this task at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

The first train swapper discovered that the bridge could be operated with at most two railcars on it. By rotating the bridge 180 degrees, the railcars switched places, allowing him to rearrange the railcars (as a side effect, the railcars then faced the opposite direction, but train railcars can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed is a routine which decides for a given train the least number of swaps of two adjacent railcars necessary to order the train. Your assignment is to create that routine.

Input

The input contains on the first line the number of test cases. Each test case consists of two input lines. The first line of a test case contains an integer L, determining the number of railcars on the train ($1 \leq L \leq 50$). The second line of a test case contains a permutation of the numbers 1 through L, indicating the current order of the railcars. The railcars should be ordered such that railcar 1 comes first, then 2, etc. with railcar L coming last.

Output

For each test case output the sentence: 'Optimal train swapping takes S swaps.' where S is an integer.

Example Input

```
3
3
1 3 2
4
4 3 2 1
2
2 1
```

Example Output

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

Problem D: Dropping Balls

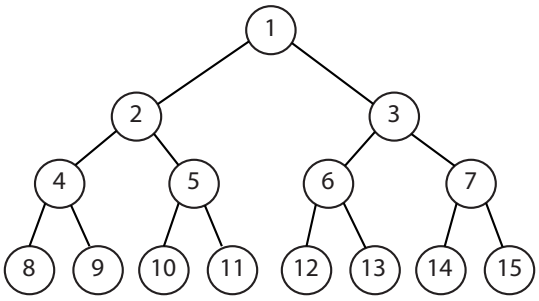
Input file: Balls.txt
Output: System.out
Time Limit: 5 seconds

A full binary tree of depth D has D levels. The root is a single node at level 1, Level 2 contains 2 nodes, level 3 contains 4 nodes, and level L contains 2^{L-1} nodes. The nodes in the bottom level are called "terminal nodes". The root is numbered 1. Nodes in the second level are numbered 2 and 3 from left to right. The numbering system continues in this fashion.

k balls are dropped one by one into the root of a full binary tree structure. The nodes of the tree contain switches that direct the balls either to the left or the right subtree of the node being visited. All switches are initialized to LEFT, so that the first ball dropped will go left at each level, ending up in the leftmost terminal node.

When a ball leaves a node, the switch in that node changes to the opposite direction, LEFT↔RIGHT.

The diagram below represents a full binary tree of depth 4. Since all of the switches are initially set to LEFT, the first ball to be dropped will change the switches at nodes 1, 2, and 4 before it finally stops at position 8. The second ball to be dropped will change switches at nodes 1, 3, and 6, and will stop at position 12. The third ball will change the switches at node 1, 2, and 5 before it stops at position 10.



A full binary tree of depth 4

You have to determine for a given tree depth, the final resting place of the k^{th} ball dropped.

Input

The first line of input will contain the number of test cases to follow. Each test case comprises a single line of text containing two space-separated integers. The first integer, D ($D < 11$) is the depth of the tree (the number of levels of nodes). The second integer, k, is the index number of the ball being dropped. The first ball has index 1, the second has index 2, and so on. k will never exceed the number of nodes in the lowest level.

Output

For each test case, give the node number of the final resting place of the ball.

Sample Input

```
5
4 2
3 4
10 1
2 2
8 128
```

Sample Output

```
12
7
512
3
255
```

Problem E: Imelda's Shoes

Input file: Shoes.txt
Output: System.out
Time Limit: 5 seconds

Imelda has n pairs of shoes in n different colors. At the end of the day she puts them back on her shoe rack, in perfect order, Lbrown,Rbrown,Lred,Rred,Lblue,Rblue. . . Each pair is together on the rack with the left shoe to the left of the right shoe.

Something mysterious happens in the night. The Shoe-Gremlin visits. He likes to try on shoes. He takes some of the right shoes off the rack and tries them on. When dawn approaches, the Shoe-Gremlin quickly puts the right shoes back on the rack in between the undisturbed left shoes. Unfortunately, the Shoe-Gremlin doesn't care about pairing up the shoes properly.

The maid doesn't want Imelda to know about the Shoe-Gremlin, so each morning, she gets up early to put the pairs together. She is very smart and realizes that she only has to swap (not necessarily neighboring) pairs of right shoes.

Your job is to write a program that tells the maid the minimum number of swaps necessary to get the shoes back as Imelda left them.

Input

The first line contains the number of test cases. Each test case consists of a single line starting with an integer giving the total number of pairs of shoes n. The following $2n$ numbers describe the initial arrangement of shoes on the rack. Each shoe is labeled by a positive integer p, $(1 < p \leq 1,000)$, where two shoes share the same label if and only if they are part of the same pair. Both the left and right shoes of a given pair will be present (remember that left and right shoes alternate).

Output

For each test case, output one line containing a single number - the minimum number of swaps needed to pair up all shoes.

Sample input

```
2
2 2 1 1 2
4 1 2 3 4 4 1 2 3
```

Sample output

```
1
3
```

Problem F: Twister Tales

Input file: Twister.txt
Output: System.out
Time Limit: 3 seconds

Dr. Jo Harding and her husband Bill are storm chasers. They have built a van to withstand F5 tornadoes.

On a series of encounters with tornadoes, the van is lifted off the ground and spun through an angle, θ degrees. Each time when they land back on the ground they aren't sure which way the van is facing (the devastation is usually so great that they have few visual markers to help them). Fortunately their compass always freezes when the tornado first hits, so they know what their heading was at that moment.

Gyros in the van tell them the counter-clockwise angle θ through which the van rotated. This angle could be negative when they chase storms in South America. You have to calculate the new heading of the van.

Input

The input begins with a single integer N, giving the number of problems to solve. Then N lines follow. Each line contains two space-separated integers giving the initial heading of the van and the counter-clockwise angle ($-100,000 \leq \theta \leq 100,000$) through which the van rotated. Both of these quantities are given in degrees. The heading is given as the clockwise angle between north and the direction in which the van was/is pointing. It is always in the range 0 to 359 inclusive.

Output

For each problem, output one line of the form, "The new heading is ###" where the number signs represent a three digit integer. Remember, a heading is given as the (non-negative) clockwise angle between north and the direction in which the van is pointing.

Sample Input

```
4
25 25
10 50
300 1000
0 -40
```

Sample Output

```
The new heading is   0
The new heading is 320
The new heading is 20
The new heading is 40
```

Problem G: Almost Palindromes

Input file: Pals.txt
Output: System.out
Time Limit: 3 seconds

A palindrome is a sequence of one or more characters that reads the same from the left as it does from the right. For example, Z, TOT and MADAM are palindromes, but ADAM is not.

Given a sequence S of N capital letters, can a palindrome be formed by deleting zero or one letter from S?

Input

The input file contains several test cases. The first line contains an integer T that indicates how many problems are to follow.

Each of the T following lines contains a sequence of letters, S, of length N, $(1 \leq N \leq 60)$, and represents one problem for you to solve.

Output

For each test case output a line of one of the forms,"A palindrome can be formed by deleting the character in position ##", "The string is already a palindrome", or "A palindrome cannot be formed by deleting a single letter". The two number signs represent a one or two digit integer in the range 1 to 60, giving the position of the first character in the string that can be deleted to form a palindrome.

Letters in the string S are numbered 1, 2, ...

Sample Input

```
4
AA
ABC
ABBBCBBBBA
ABBA
```

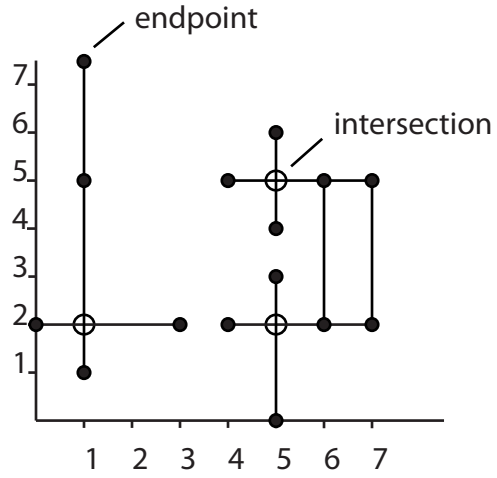
Sample Output

```
The string is already a palindrome
A palindrome cannot be formed by deleting a single letter
A palindrome can be formed by deleting the letter in position 5
The string is already a palindrome
```

Problem H: Intersections

Input file: Crossings.txt
Output: System.out
Time Limit: 3 seconds

You are given the endpoints of N line segments in the x-y plane. All the line segments are either horizontal or vertical. The endpoints have integer coordinates in the range (0,0) to (100,100) inclusive. You have to report all the intersections between line segments. See the diagram below.



Line segments corresponding to the first test case below

An intersection is a point shared by a horizontal and a vertical line segment and is not an endpoint of either line segment.

Input

The first line of text in the input file contains a single integer P, giving the number of test cases to follow. P test cases follow.

The first line of text of each test case contains one integer, N, ($N < 100$), that gives the number of line segments. Then N lines of text follow, each one containing four space-separated integers, x_1, y_1, x_2, y_2 corresponding to a horizontal or vertical line segment (p_1, p_2)

There will not be any pair of line segments that share more than one x-y point.

Output

For each test case, if there are no intersections, print the line "No crossings". Otherwise list all the intersections, one per line in order of increasing x coordinate. Each intersection will be printed as two space-separated integers. If more than one intersection occurs at the same

x coordinate, list them in order of increasing y value. Follow the output records for each test case by a blank line.

Sample Input	**Sample Output**
3 9 1 1 1 5 1 5 1 7 0 2 3 2 4 2 7 2 6 5 6 2 4 5 7 5 7 2 7 5 5 3 5 0 5 6 5 4 2 0 2 10 2 0 1 10 1 2 5 0 5 10 0 5 10 5	1 2 5 2 5 5 No crossings 5 5