8th Gulf Programming Contest

March 21,22 2018

Zayed University

Dubai, UAE

Duration: 10 am - 3 pm

Problem Set

Problem #	***Problem Name***	***Balloon Color***
A	Rocket Fuel	Light Blue
B	Socks of The Dark	Red
C	Squaring the Sequence	Black
D	Flu Epidemic	White
E	Chain of Coins	Orange
F	Inherited Land	Dark Blue
G	The Fantasy League	Golden
H	Rotating keyboard	Green
I	Fermat's Triangular Numbers	Yellow
J	Flight Duration	Pink
K	The Gabriel Graph	Purple

A. Rocket Fuel

Program:	rocket.(cpp|java)
Input:	rocket.in
Balloon Color:	Light blue

Description

A new material is discovered and can be used as fuel to send rockets into far space. This material however, has unique characteristics. It can thrust the rocket in acceleration such that the speed increases from 0 to the maximum speed, then deaccelerate gradually from maximum speed to 0, when the rocket lands at target planet. Every time the rocket speed is increased or decreased into a new speed value *S,* the rocket consumes *S* grams of fuel from this material. The rocket starts at the speed of 0, then 1 unit, and after that every step it increases by 3 until it reaches the maximum required speed *n*, then it immediately starts decreasing by 3 until it reaches 1, then 0 for landing. Rocket scientists calculate the maximum speed the rocket must reach in order to land correctly at the target planet, and your task is to calculate the amount of fuel needed in grams.

Example

In order to send a rocket to the moon, n = 16, so the speed changes as follows:

0 1 4 7 10 13 16 13 10 7 4 1 0

Fuel needed = 0 + 1 + 4 + 7 + 10 + 13 + 16 + 13 + 10 + 7 + 4 + 1 + 0 = 86 grams.

Input

The input consists of several sequences of numbers terminated by 0, each one on a separate line and represents the maximum rocket speed, where $4 \le n \le 4000000000$, note that $n = 3*k + 1$, where $k = 1, 2, 3, \ldots$. Input is terminated by a sequence having $n = 0$, which should not be processed.

Output

For each sequence, you are to output one line, containing the required fuel in grams.

Sample Input / Output

rocket.in

```
4
7
22
16
0
```

OUTPUT

```
6
17
162
86
```

B. Socks of The Dark

Program:	socks.(cpp|java)
Input:	socks.in
Balloon Color:	Red

Description

You're in trouble! The lights in your bedroom are out and the room is completely dark. Your mission, should you choose to accept it, is to get a pair of socks. You're not much of an organized person, so your sock drawer is a mess and socks are spread around randomly. You've got a lot of different colored socks, and what you need is a pair that has the same color.

In this problem, you are given the number of different colors, followed by the number of socks you have of each color. Your task is to calculate the minimum number of socks you should pick out of your drawer, given that you cannot see what color they are, so that when you leave the room you would have at least two that are of the same color.

Input Format

The input starts with a number T ($1 \le T \le 1{,}000$) that represents the number of test cases in the file. Each test case starts with a line that contains a single integer, which is the total number of colors C ($1 \le C \le 1{,}000$). The following line contains C integers, each representing the number of socks of a unique color C_i ($1 \le C_i \le 10^9$).

Output Format

The output for each test case is in this form:

```
k. M
```

where ***k*** represents the test case number (starting at 1), and ***M*** is the maximum number of socks as described. ***M*** is "Impossible" without the quotes, if no matching socks exist.

Sample Input / Output

socks.in

```
2
2
2 2
2
1 1
```

OUTPUT

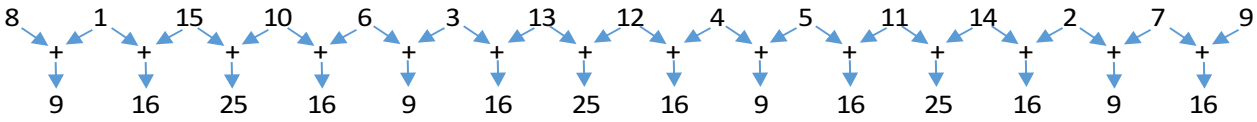
```
1. 3
2. Impossible
```

C. Squaring the Sequence

Program:	`square.(cpp|java)`
Input:	`square.in`
Balloon Color:	`Black`

Description

A square number is one that has an integer square root, e.g. 4, 9, 16, are square numbers. It was discovered that the numbers from 1 to 15 can be ordered in such a way that any two successive ones add-up to a square number:



Such an ordering is neither unique, nor necessary in general. For example, the numbers from 1 to 18 cannot form such a sequence.

Your task is to find such a sequence for an arbitrary set of integers provided to you. If multiple solutions exist, you are supposed to provide the one that starts with the smaller numbers possible, or output "No solution" is such an ordering is impossible.

Input Format

The input has a number of test cases (<100). Each test case starts with a line containing an integer N ($1 \le N \le 40$) that declares the integers making up our set. N numbers follow in the next line, separated by white space. The numbers are positive integers ≤40. The input ends with an N equal to 0.

Output Format

For each test case you should output in a single line a sequence where successive numbers form square sums, or

```
No solution
```

if such a sequence is impossible. If multiple sequences exist, you should output the one that starts with the smallest numbers.

Sample Input / Output

square.in

```
10
2 2 2 2 2 2 2 2 2 2
10
3 3 3 3 3 3 3 3 3 3
15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0
```

OUTPUT

```
2 2 2 2 2 2 2 2 2 2
No solution
8 1 15 10 6 3 13 12 4 5 11
14 2 7 9
```

D. Flu Epidemic

Program:	flu.(cpp|java)
Input:	flu.in
Balloon Color:	White

Description

The region is facing a flu epidemic. And it is a big problem as it is spreading fast and incapacitating the victims for a large time T (i.e. people stay ill for time T since they become infected).
For someone to get infected, they have to be within distance R from an existing patient as the virus travels by air. The virus becomes active and spreads to other people, after an incubation period I.
However, apart from staying at home, there is little someone can do to prevent the spread of the disease. But it does present an opportunity for studying infectious decease behavior in crowded places.
In order to validate the results of your research, you need to predict how many people become infected when they occupy the same room for some time duration D. You are thankfully supplied with the positions of the people in the room.
For example, in the following room occupied by 10 people (cells containing "P" represent healthy people and "I" represent the infected), after 20 minutes 3 people will be ill, if T=1000 minutes, I=20min and R=3 meters. After 40minutes the number will rise to 4:

State at time t=0

	0	1	2	3	4	5	6	7
0	P							P
1		P						
2			P	I				
3								
4								
5								
6	P							P
7				P	P		P	

State at time t=20

	0	1	2	3	4	5	6	7
0	P							P
1		I						
2			I	I				
3								
4								
5								
6	P							P
7				P	P		P	P

State at time t=40

	0	1	2	3	4	5	6	7
0	I							P
1		I						
2			I	I				
3								
4								
5								
6	P							P
7				P	P		P	P

Distances between people are calculated based on the Euclidian distance of the centers of the cells they occupy. Each cell is assumed to be 1meter x 1meter in size. In the example shown above, the distance between the initially infected person and the top left corner is $\sqrt{3^2 + 2^2} = 3.6m$ and that is why the top left person is infected after 40 minutes by the newly infected ones.

We can assume that all the initially ill people, contracted the virus just before getting in the room. Also if a person is cured during the time period examined (i.e. T<D), he/she is not infected again. If an infected person is cured at time D+δt (i.e. immediately after D), it still counts as an ill person.

Input Format

The input file contains several test cases. Each test case starts by a line declaring the width W and height H of the room in meters (both integers <100). The following line lists the T, I, R and D parameters separated by whitespace (all are positive integers with I < T). H lines follow, each containing W characters. Each character can be either a 'P' for a person, a 'I' for infected, or 'X' for empty space.
The input ends with a pair of zeros for W and H.

Output Format

For each test case you should output the number of ill people in the room at time D.

Sample Input / Output

flu.in

```
8 8
1000 20 3 60
PXXXXXXP
XPXXXXXX
XXPIXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
PXXXXXXP
XXXPPXPX
0 0
```

OUTPUT

```
4
```

E. Chain of Coins

Program:	`chain.(cpp|java)`
Input:	`chain.in`
Balloon Color:	`Orange`

Description

Khalid is selling chains of coins that has Arabic numerals inscribed on them for tourists. Every chain is composed of one or more coins with an inscription of only one digit inside. Every tourist is looking for his own chain to buy and is willing to pay a specific price to get it. The following rules are used:

- Khalid has one chain with two ends; a portion can be extracted from either end or from the middle.
- When Khaled extracts a portion from the middle of the chain, he joins the remaining left and right parts at the place where he removed the portion.
- Every tourist has one favorite chain of one or more digits, with a specific price to pay.
- Every tourist is willing to have as many chains of his favorite sequence as available.
- Any portion removed from the chain must be sold, i.e. Khalid cannot remove anything from the chain and throw it or reattach it.
- Once extracted, the chain can be reversed, i.e. 109 = 901.
- Khalid does not have to serve all tourists, nor to serve them in any particular order.
- Khalid knows all the chains of the tourists together.

Your task is to tell Khalid what is the maximum profit he can make out of his chain.

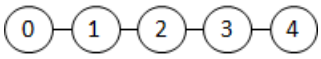
Example

Given the main chain below:

Five tourists have the following requests for their chains followed by the price they'll pay:

```
01: 5
109: 8
91: 4
13: 2
210: 9
```

Solution: 10, 10, 91, 13 for a total of 5 + 5 + 4 + 2 = 16.
Note the other option: 109, 10, 13 will result in 8 + 5 + 2 = 15.



Three tourists have the following chains followed by price

```
0123: 10
123: 6
40: 5
```

Solution: 123, 40, for a total of 5 + 6 = 11.
Note the other option: 0123 will result in 10.

Input

The input file starts with a positive integer *k* representing the number of coins in the chain, followed by *k* digits that are not separated by spaces on the next line. On the following line is a single integer *n* that represents the number of tourists, followed by *n* lines, each having two integers, *C* and *P*. *C* represents the digits in the tourist's chain and *P* represents the price he/she is willing to pay. Note that:

$1 \le k \le 30$
$1 \le n \le 10$
$1 \le length(C) \le 9$
$1 \le P \le 100$

k=0 indicates the end of the input.

Output

For each problem instance, your program should output one line that contains the maximum price Khalid can sell chains for.

Sample Input / Output

chain.in

```
11
10912110374
5
01 5
109 8
91 4
13 2
210 9
5
01234
3
0123 10
123 6
40 5
0
```

OUTPUT

```
16
11
```

F. Inherited Land

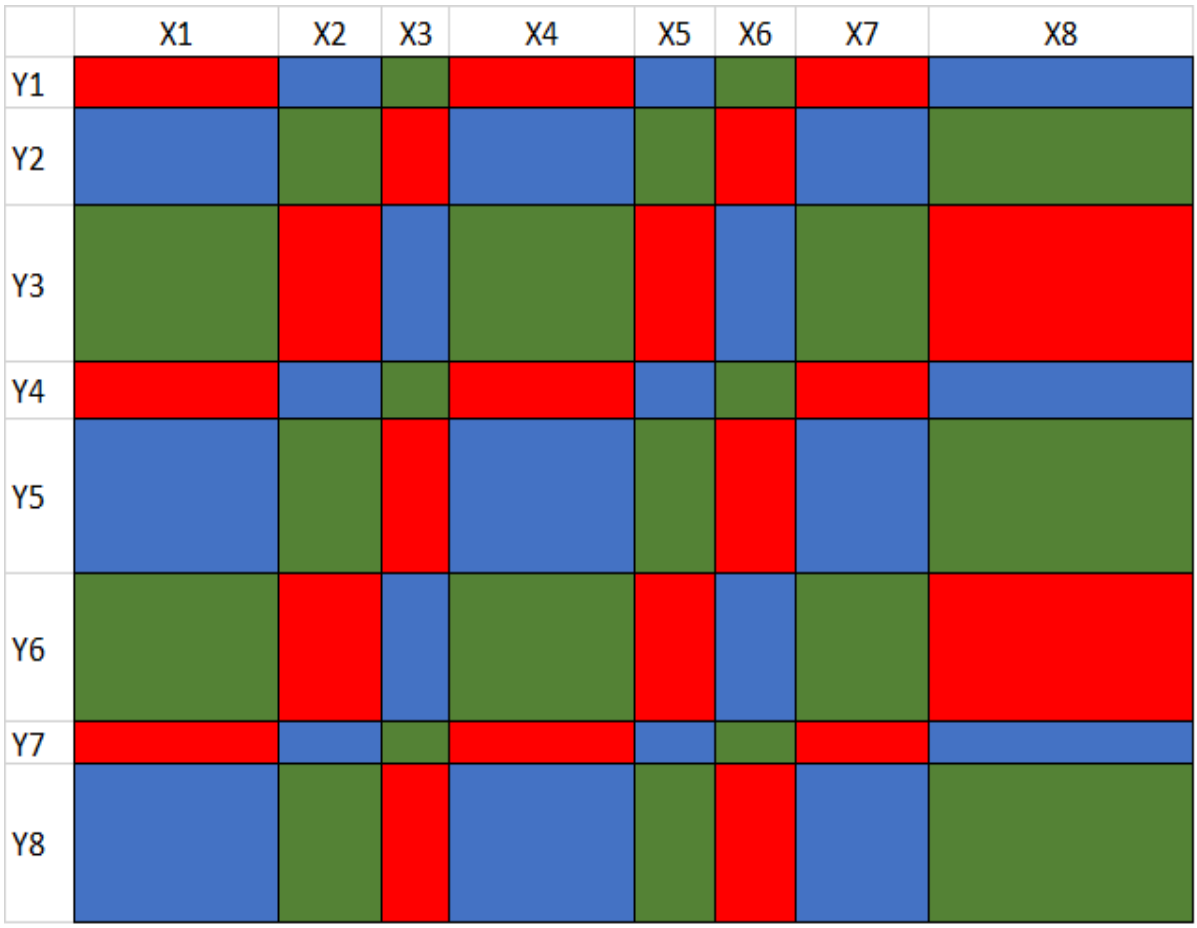
Program:	`land.(cpp|java)`
Input:	`land.in`
Balloon Color:	`Dark Blue`

Description

King Color died and left 3 princes behind: Red (22 years old), Green (21), and Blue (20). They inherited a huge piece of land with rectangular shape; the three children had a dispute on how to divide the land between them, until their father's advisor suggested to use the following method:

He partitioned the land into *n* vertical stripes of random lengths: X1, X2, … Xn, meters and *n* horizontal stripes of random lengths Y1, Y2, …, Yn meters.

These stripes split the land into *n x n* rectangles. The intersection of vertical stripe *i* and horizontal stripe *j* has code number *(i + j) mod 3*, and hence is given to the prince with the same first digit in his age. For example area X1 Y1 has the code *(1+1)%3 = 2*, hence it is given to prince Red (22), area Y6 X4 has the code 1 hence it is given to prince Green (21). See the figure for *n = 8*. Your task is to calculate the area of land given to each prince.



Input

The input consists of several problem instances, each starts with the integer *n* on the first line, followed by two lines each has *n* integers for the values of X1, … , Xn, separated with single spaces, followed by the values of Y1, … , Yn on the next line separated with single spaces. Then next problem instance follows starting with a new *n*. Problem instances terminate when *n = 0*. Problem boundaries are:

$3 \leq n \leq 250000$
$1 \leq Y_i, X_i \leq 10$

Output

For each problem instance print out the area of land for the three princes in order: Red Green Blue separated by a single space, one line for each problem instance.

Sample Input / Output

land.in

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

OUTPUT

```
3 3 3
19 16 19
197 131 155
```

G. The Fantasy League

Program:	fantasy.(cpp|java)
Input:	fantasy.in
Balloon Color:	Golden

Description

In Fantasy League, players are given a number of points based on the number of goals, assists, saves, goals conceded, and other factors. Each player has an associated cost. Your task in the game is to form a team with the best players, i.e. the players with the highest total score. However, you're limited with a specific budget, so the total cost of players in your team should not exceed your budget. You also need a specific number of players in your team, meaning you cannot have more or less players than specified.

In this problem, you are given a budget, the specific number of players your team should have, and the number of players available to choose from. This is followed by the number of points and the cost of each respective player. Your task is to calculate the maximum number of points your optimal team can achieve, while remaining within budget and having the required number of players.

Input Format

The input starts with a number T ($1 \le T \le 1{,}000$) that represents the number of test cases in the file. Each test case starts with a line that contains three integers B ($1 \le B \le 10{,}000$), N ($1 \le N \le 100$), K ($1 \le K \le N$), representing the budget, the number of available players, and the number of players the team should have, respectively. N lines follow, with each containing an integer P_i ($1 \le P_i \le 100{,}000$) representing the number of points achieved by the ith player, followed by a single decimal number C_i ($0.1 \le C_i \le 1{,}000.0$) representing the cost of the ith player.

Output Format

The output for each test case is in this form:

```
k. M
```

where ***k*** represents the test case number (starting at 1), and ***M*** is the maximum number of points your optimal team can achieve, with the given restrictions. ***M*** is "Impossible" without the quotes, if a team cannot be formed with the given requirements.

Sample Input / Output

fantasy.in

```
2
100 3 2
10 30.0
15 70.0
20 70.0
50 3 2
20 50.1
10 60.2
5 70.3
```

OUTPUT

```
1. 30
2. Impossible
```

H. Rotating keyboard

Program:	`keyboard.(cpp|java)`
Input:	`keyboard.in`
Balloon Color:	`Green`

Description

You have a strange cellphone. The cellphone screen has two parts. The upper part shows the number that is typed. The lower part displays the digits and symbols and a cursor. However, the screen is not a touch screen. Therefore, the only way to type a digit is to attach a *detachable keyboard* to the cellphone screen, which consists of only four arrow keys and an OK button. The function of the arrow keys is only to move the cursor on the screen, without outputting anything. The function of the OK button is to output the digits on which the cursor is currently located. The initial position of the cursor is on the digit 5.

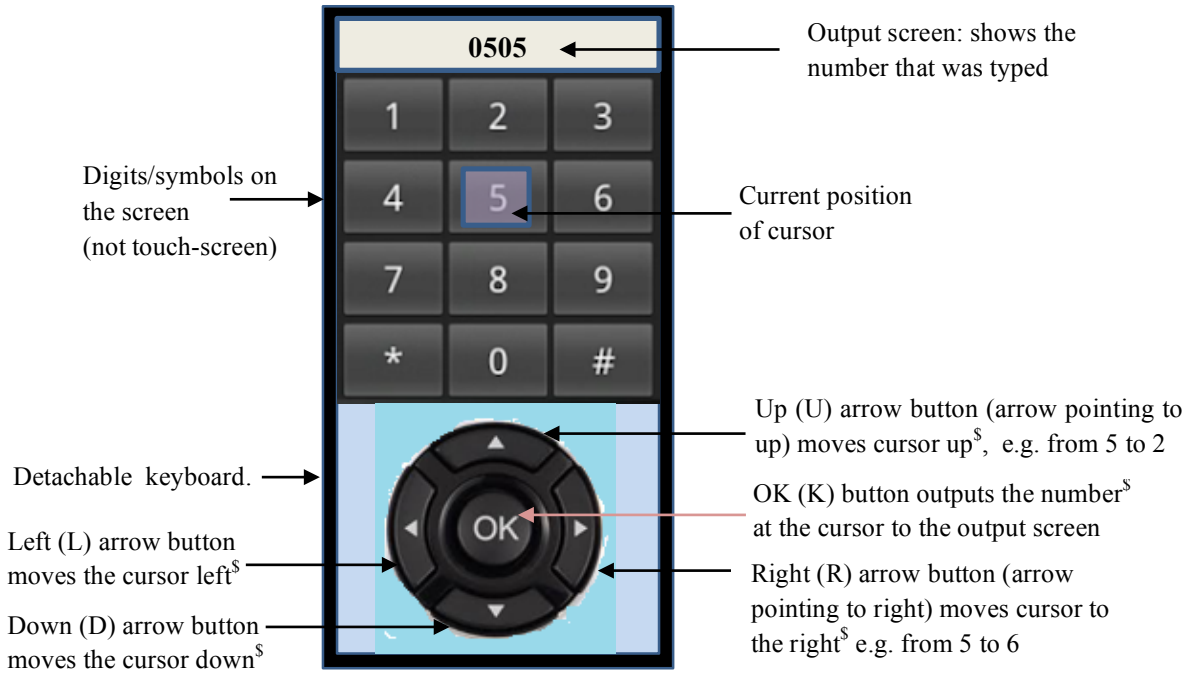


Figure H.1: The cellphone screens and keyboard. The shorthand names of the arrow keys will be L, R, U, and D corresponding to the arrow keys pointing to Left, Right, Up and Down. The OK key will be shorthanded as K. The number 0505 on the output screen was typed by the following key sequence (assuming initial cursor position on 5): DDKUUKDDKUUK.

$ see the terms and conditions next page.

So, for example, if someone wants to type the digit 0 (starting at the initial state, i.e., at button 5), he has to press the following key sequence: DDK. The first D will move the cursor down to 8, the second D will move it to 0, and then K will type 0. After typing the digit, the cursor location will remain on the digit 0. Given the initial position of the cursor at button 5, and a sequence of key-presses, you are to find the number (consisting of one or more digits) that was typed.

Wait, there is a catch! The detachable keyboard can be attached by rotating 90, 180, or 270 degrees clockwise. However, even after rotation the function of each arrow key remains the same, for example, the key that moved the cursor down before rotation, will still move the cursor down after

rotation. But according to the user (i.e., observer), the Left arrow key at the rotated position is different from the Left arrow key before rotation. As an example, if the keyboard is rotated 90 degrees, it will look as follows:

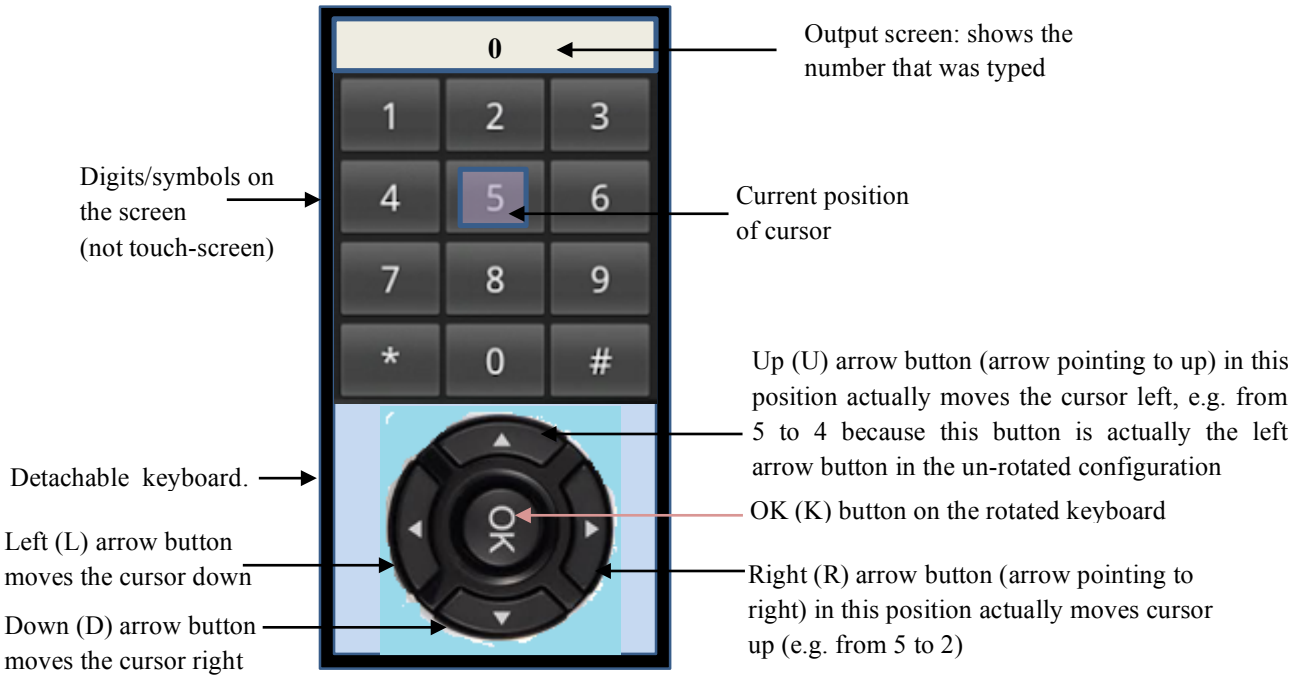


Figure H.2: The cellphone screens and the 90 degree clockwise rotated keyboard. In this case, the Left (L) arrow button corresponding to the observer is the button pointing to Left (i.e., to West). But actually it is the Down button of the un-rotated keyboard. Therefore, pressing the sequence LLK will produce the output 0. Because the first L will move the cursor Down to 8, second L will again move the cursor Down to 0 and K will output 0.

[$]**terms and conditions (read carefully)**: The cursor will *not* move if it is instructed to move to an invalid position (e.g. cursor is at 0 and instructed to move down, it will not move). Also, if the cursor is on the * or # symbol, then it will ignore the OK command, i.e., pressing K will not type the symbol.

Input

The input consists of n (0< n <= 100) cases. Each case starts with two integers r <space> s <space> where r ∈ {0, 90, 180, 270} is the degree of clockwise rotation of the keyboard, and s (<= 10) is the number of scenarios. Then s lines follow, where each line consists of a scenario of the key sequence (max length 50) that was pressed by a user. Assume that each scenario starts with the cursor position at digit 5, and there will be at least one digit output for any scenario.

Output

For each case, you are to output the case number t in one line, then for each scenario, print in one line the number that was typed, without any leading or trailing spaces.

Sample Input / Output

keyboard.in

```
2
0 1
DDKRKRUKUKUK
90 1
DDKRKRUKUKUK
```

OUTPUT

```
1
0963
2
63211
```

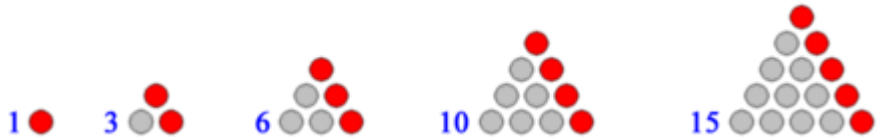
I. Fermat's Triangular Numbers

Program:	tri.(cpp|java)
Input:	tri.in
Balloon Color:	Yellow

Description

A Triangular Number is a number that can be represented as dots or pebbles arranged in the shape of an equilateral triangle.

Here are the first 5 triangular numbers (source: Wikipedia)



Fermat's Triangular number theorem (proven later by Carl Gauss) states that every positive integer is the sum of at most three triangular numbers. For example, 17 (which isn't a triangular number) is the sum of 1+6+10 (all being triangular numbers). Here are a few more examples:

2 = 1+1
17 = 1+6+10
49 = 21+28

Given an integer *n* determine the triangular numbers that sum up to it.

Input Format

The input starts with a number **T** representing **T** test cases in the file. Every test case is described on a single line with a single integer **N** (**1<=N<=100,000,000**).

Output Format

For each test case print one line of output in the form:

```
k. result
```

where **k** is the test case number (starting at 1), and **result** is the list of triangular numbers that sum up to the integer ***n*** for test case **k** in an increasing order, separated by a single space. If more than one solution exists print the one with the least number of triangular numbers. If the expression has the same number of triangular numbers, print the one with the smallest triangular number. If both expressions have the same smallest triangular number, print the one with the second smallest.

For example, 19 = 1+3+15 or 3+6+10. Your program should output the first solution. If the number *n* is triangular, then you should output the number itself.

Sample Input / Output

tri.in

```
6
10
19
16
17
48
49
```

OUTPUT

```
1. 10
2. 1 3 15
3. 1 15
4. 1 1 15
5. 3 45
6. 21 28
```

J. Flight Duration

Program:	flight.(cpp|java)
Input:	flight.in
Balloon Color:	Pink

Description

I have always had a problem with time! No, hold your horses, not reading time, I can read a clock (both analog and digital) very well, thank you! But rather figuring out a flight's duration when the origin and destination are on two different time zones. Will you help me out?

Write a program that reads a flight's take off time, it's time zone, and landing time and the destination's time zone, and calculates the flight's duration.

Input Format

The input starts with a number **T** that represents the number of test cases in the file. Each test case is described on single line of the following form:

hh_1:mm_1 dd_1GMT hh_2:mm_2 dd_2GMT

where **hh_1:mm_1** represents the time of departure in a 24 hour format, and **dd_1** (-12<=**dd_1**<=+12) is the time zone of the flight's origin. Similarly, **hh_2:mm_2** and **dd_2** are to describe landing time, and destination's time zone.

Output Format

For each test case print one line of output in the form:

k. h:mm

where **k** is the test case number (starting at 1), and **h:mm** is the time the flight takes. Note that flight duration will always be less than 24 hours.

Sample Input / Output

flight.in

```
2
09:05 +03GMT 13:30 +00GMT
20:00 +02GMT 04:00 -02GMT
```

OUTPUT

```
1. 7:25
2. 12:00
```

K. The Gabriel Graph

Program:	gabriel.(cpp|java)
Input:	gabriel.in
Balloon Color:	Purple

Description

The Gabriel graph is the graph with vertex set V in which any points a and b are adjacent if they are distinct and the closed disc of diameter d(a, b) contains no other elements of V. d(a, b) is the distance, e.g., Euclidean distance, between the two points a and b. The result of applying a Gabriel graph on a set of points is illustrated in Figure 1.

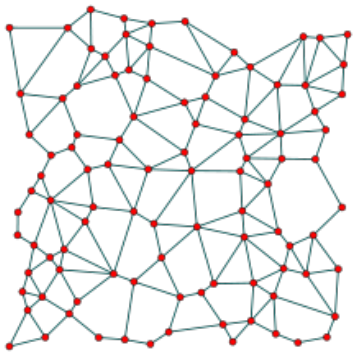


Figure 1. Example of a Gabriel graph on a set of points.

The following figures provide examples of the verification of the property of Gabriel graph and another with that property is not verified.

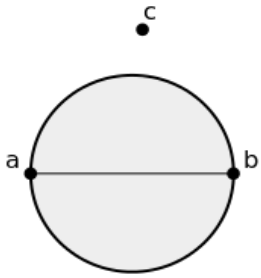


Figure 2. Points a and b are neighbors because c is outside the disc of diameter d(a, b).

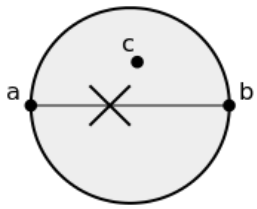


Figure 3. Points a and b are not neighbors because c is inside the disc of diameter d(a, b). If c is on the disc, then a and b are still not neighbors.

Your task is given a set of 2D points, to find the corresponding Gabriel graph by calculating the neighbors of each point.

Input format

The input starts with a number X ($1 \leq X \leq 100$) that is the number of test cases in the file. Each test case starts with 2 numbers: Y (the test case number) and Z ($2 \leq Z \leq 10$) that is the number of points to be examined. Z lines follow, each containing three numbers separated by whitespace: the point number and its x and y coordinates (e.g., 1 3 4: node 1 has 3 as x coordinate and 4 as y coordinate). The x and y coordinates are 32-bit integers.

Output format

For each test case, you should output the test case number, followed by Z lines, one for each point in the test case. In each line, you should output the node number along with the numbers of the nodes that are its neighbors. Neighbor numbers should be sorted and they should be separated by a single space.

Sample Input / Output

gabriel.in

```
1
1 10
1 2 3
2 4 5
3 6 7
4 8 9
5 1 3
6 6 5
7 2 1
8 4 1
9 3 4
10 1 2
```

OUTPUT

```
1
1 5 7 8 9 10
2 3 6 9
3 2 4 6
4 3
5 1 10
6 2 3 8
7 1 8 10
8 1 6 7 9
9 1 2 8
10 1 5 7
```