A

Problem A

A Game with Grandma

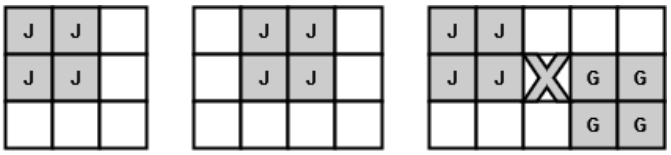
Jhinuk and her grandma have a special bond, they love playing puzzle games together. One day, grandma came up with a new game for Jhinuk to play. She found some old wooden blocks and put them together in a **3xN** grid.

To make the game more fun, grandma made up some imaginary rules: two players take turns picking a **2x2** section of the grid to color, with the rule that each cell can only be colored once. The player who can no longer pick a **2x2** section to color loses the game.

The game sounded easy to Jhinuk, but there was a catch. Some of the blocks were cracked and couldn't be painted!

Jhinuk and her grandma are both smart and play well, so they will never make a wrong move if there is a way to win. Can you write a code to figure out who will win given the starting state of the board?

Grandma always lets Jhinuk go first 'cause she loves her so much. To help you write the code, she has provided you with some sample boards.



Input

The first line will contain a single integer **T (1 ≤ T ≤ 300)**. The first line of each test case contains **N (1 ≤ N ≤ 100)** representing the size of the grid. Each test case will contain 3 lines of the same length, representing 3 rows of blocks. Each block is represented by a character “O” or “X” indicating a good block or a damaged block respectively.

Output

For each case, print the case number and the winner of the game - “Jhinuk” or “Grandma”.

Sample Input	Output for Sample Input
3 3 OOO OOO OOO 4 OOOO OOOO OOOO 5 OOOOO OOXOO OOOOO	Case 1: Jhinuk Case 2: Jhinuk Case 3: Grandma

B Problem B

Transform the Array

Alice and Bob, each of them are given two arrays of positive integers, **A** and **B** of equal lengths. They need to transform array **A** into array **B** by performing some operations on the numbers.

Alice can perform the following operations:

Alice's operations:
Alice can choose a number D where D is the divisor of one of the numbers in the array **A** and perform one of the following two operations:

- Multiply one of the numbers in array A by D
- Divide one of the numbers in array A by D. This operation can only be performed if the division result is an integer.

Alice can repeat these operations as many times as she wants and can keep modifying the array A.

Bob can perform the following operations:

Bob's operations:
Bob can choose a number **D** where **D** is a divisor of all the numbers of the array **A** and perform one of the following two operations:

- Multiply all the numbers of the array **A** by **D**
- Or, divide all the numbers of array **A** by **D**. Note, this operation will always result in integer numbers.

Bob can repeat these operations as many times as he wants and can keep modifying the array A.

Now, Alice and Bob need to tell us if they can transform array A into array B using their allowed operations. Please note, Alice can not use Bob's operation and similarly vice versa.

Input

The first line of the input will contain an integer **T (1 ≤T ≤ 10000)** describing the number of test cases.
The first line of each test case will contain an integer **N (1 ≤ N ≤ 500000)** denoting the length of the arrays.
Next line of the test case will have N numbers of array **A (1 ≤ A_i ≤ 10000000)**. The following line will contain N numbers of array **B (1 ≤ B_i ≤ 10000000)**.
The sum of N from all the test cases in the input file will not exceed 500000.

Output

The output of each test case should start with the case number in the format "**Case X:**" (without quotes) where **X** is the test case number following two strings **P** and **Q** separated by exactly one space.

The value of P will be "**Yes**" (without quotes) if Alice can transform array A into array B using her allowed operations, otherwise "**No**" (without quotes). Similarly, the value of Q will be "**Yes**" (without quotes) if Bob can transform array A into array B using option 2, otherwise "**No**" (without quotes). Please see the sample output for clarity.

Sample Input	Output for Sample Input
1 2 5 14 7 5	Case 1: Yes No

Explanation:

The given array A is [5, 14] and array B is [7, 4].
While processing option 1 for transformation:

- Multiply 5 by 7: [35, 14]
- Multiply 14 by 5: [35, 70]
- Divide 35 with 5: [7, 70]
- Divide 70 with 14: [7, 5]

So, the first output is Yes

For option 2, we have only one common proper divisor for both of the numbers and that is 1, so we have no way to transform the array A to array B using the option 2. That's why the answer is No.

Note: You may have to use a faster I/O method.

C Problem C

Omago Virus

A new variant of the COVID-19 virus has been detected in the country. This variant is called the **'omago'** variant because once someone becomes infected with the variant, they will always be infected. The new variant is also very contagious. Scientists wonder how many days it will take for the virus to infect everyone in the country.

There are **n** people in the country. Initially, **m** people are infected with the virus. Every day each infected person spreads the virus to **k** randomly chosen distinct people except himself. If the virus spreads to a non-infected person, they become infected at the beginning of the next day. An infected person will always be infected. Find the expected number of days before every person becomes infected.

Input

The first line will contain a single integer **T**, the number of test cases.
Each case is described by a single line containing three space-separated integers **n, k,** and **m**.

Output

For each case, print a single integer in a new line, the expected number of days before everyone becomes infected modulo **998244353**.

Formally, let **M = 998244353**. It can be shown that the answer can be expressed as an irreducible fraction **p/q**, where **p** and **q** are coprime integers and **q** is not divisible by **M**. You should output the integer equal to $p \cdot q^{-1} \pmod{M}$. In other words, output an integer **x** such that $0 \le x < M$ and $q \cdot x \equiv p \pmod{M}$

Constraints

- $1 \le T \le 11$
- $1 \le n \le 400$
- $1 \le k, m < n$
- **n ≤ 100 for at least T - 1 cases.**

Sample Input	Output for Sample Input
2 3 1 1 5 4 2	332748120 1

For the first case, the expected number of days is **7/3**,
So the answer is $7 \cdot 3^{-1} \pmod{998244353} = 332748120$.

For the second case, initially 2 people are infected. On the first day, both persons will spread the virus to all other persons. Thus it will take only 1 day.

D

Problem D

Omicron Juice 2023

Here I am again with my special Omicron Juice! Here goes the story: I have three kids- Alpha, Beta and Gamma. Every morning I need to prepare them for school, serve them breakfast and say them bye. You might think that I don't like all these. But to be honest I do like it, specially I have fun serving them breakfast. Every morning I serve them Omicron Juice. But I can't always pour the juice in equal amounts. If they don't get equal amounts of juice then they start fighting. I use two other glasses to make the juice amount equal. To be more specific…

There are three glasses with juice of amount A, B and C. I have two other glasses which can hold exactly K amount of drinks each and initially empty. Let's call these two extra glasses as cups to avoid ambiguity. In one operation, I pour K amount of drinks from one of the three glasses to my first cup, and from one of the two remaining glasses to my second cup. Next I pour the drink from my first cup to one of the three glasses and from my second cup to one of the remaining two glasses. Of course this move is invalid if we attempt to pour less than K amount of drinks. If I tell you A, B, C and K, can you figure out if they will go to the school peacefully or are they going to start a fight? I promise if there is any possibility for a peaceful morning I will definitely make that happen!

Just to clarify, the morning would be peaceful if all those three glasses contain equal amounts of juice, the extra **K** unit cups are empty and during the entire process, you do not spill any juice. Also, you may assume that the kids' glass can hold any amount of juice. And I can perform the operation as many times as I wish.

Input

The input begins with the number of cases **T (1 ≤ T ≤ 1,000)**. In each of the following **T** lines, you will find four non-negative integers: **A, B, C,** and **K** (each of at most **2,000,000**). **K** is a positive integer.

Output

For each case, output the case number and print if the morning is "**Peaceful**" or if they are going to have "**Fight**". Please see the sample for details.

Sample Input	Output for Sample Input
2 3 5 1 2 3 9 6 2	Case 1: Peaceful Case 2: Fight

E Problem E

Cyclic Palindrome

You are given a string $S_1S_2...S_n$ of length **n**, consisting of both lowercase and uppercase English alphabets and '?'. The beauty of a string, **beauty(S)** is equal to the number of ways to replace those '?' with lowercase English alphabets such that the string will be a palindrome.

There are **n** cyclic shifts of the string (starting from 0 to n - 1). The 0^{th} shift of the string **S**, **shift(S, 0)** is the initial string itself. The i^{th} shift of the string **S**, **shift(S, i)** is $S_{n-i+1}...S_nS_1S_2...S_{n-i}$ and so on. The score of the string **S** is given by the following formula:

$$score(S) = \sum_{0}^{n-1} beauty(shift(S, i))$$

You have to determine the score of the string S. Since, the score can be large, output score(S) modulo **1000000007**.

Input

The first line will contain a single integer **T**. Each test case will have a string **S** containing lowercase and uppercase English letters and '?'.

Output

For each case, print one line with "**Case <x>: <y>**", where **x** is the case number, **y** is score(S) modulo 1000000007.

Constraints

- **1 ≤ T ≤ 100000**
- **1 ≤ |S| ≤ 500000**
- **1 ≤ Sum of |S| over all test cases ≤ 500000**

Sample Input	Output for Sample Input
2 a?B?a abc	Case 1: 54 Case 2: 0

F Problem F

Proximity Card Data Statistics

University S has collected data from the students for the new proximity card that the students will need to enter the new permanent campus. The time window for entering the data was 1st to 5th February 2023. You will be given the data and you will have to prepare some statistics for them.

Input

First line of the input file contains a positive integer **N (0 < N < 1200)** which denotes the total number of lines to follow. Each of the next N lines contains information entered about a single student.

Each of these **N** lines starts with a timestamp of the entry which has the format **`day/month/year#hour:minute:second`** and it denotes the time of the entry. Each of these values are given as a two digit number but the year is entered as a four digit number. This timestamp is followed by the official email address of the student (Can have maximum **25** characters), this address is unique for each student as the first part of the email address is student ID and that is why a student cannot have more than one official email address. You can assume that this email address is a valid email address. This email address is followed by the birthdate of the student which has the format day/month/year, again here the day and month are given as two digit positive integers and year is given as a four digit positive integer. You can assume that the given date is valid and the year is within the range **1000** to **2023** (Inclusive). The date of birth is followed by another string which denotes the blood group of the student. This string will have any one of the following values “A+”, “A-”, “AB+”, “AB-”, “B+”, “B-”, “O+”, “O-” (without the quotes). **More than one line of input can be entered by the same student (Having the same email address), in that case only the one that was entered last (Decided by the timestamp and not by the order in the input file) will be counted.** You can assume that timestamps of more than input from the same student will be different.

Output

For the given input you should produce the number of students in each blood group: In eight different lines the blood group name should be followed by a space and then the count of students having that blood group. The blood group names should appear in lexicographic order (As shown in the output for sample input). Each of the next few lines should contain two integers, first one denoting a year and the second one a positive integer that denotes how many students were born in that year. Only the years where at least one student was born should be reported and yearly count should be reported in ascending order of year.

Sample Input

```
12
01/02/2023#15:58:31 2022100011006@seu.edu.bd 28/07/2001 A+
01/02/2023#17:46:00 2022100000101@seu.edu.bd 13/03/2003 A+
01/02/2023#18:29:15 2019100010035@seu.edu.bd 09/10/1994 A+
01/02/2023#20:09:54 2021200000054@seu.edu.bd 18/07/2001 A+
01/02/2023#22:47:33 2021100000021@seu.edu.bd 03/10/2002 A+
04/02/2023#20:10:10 2020000000014@seu.edu.bd 04/05/2001 A+
01/02/2023#10:40:16 2018200010061@seu.edu.bd 15/11/1997 A-
05/02/2023#11:21:51 2022100000067@seu.edu.bd 02/11/2002 AB+
05/02/2023#11:46:17 2018100011010@seu.edu.bd 16/12/1997 AB+
01/02/2023#23:20:28 2018100000041@seu.edu.bd 15/06/2000 B-
02/02/2023#21:58:11 2021200010017@seu.edu.bd 15/06/1997 B-
02/02/2023#21:59:11 2021200010017@seu.edu.bd 15/06/1997 B+
```

Output for Sample Input

```
A+ 6
A- 1
AB+ 2
AB- 0
B+ 1
B- 1
O+ 0
O- 0
1994 1
1997 3
2000 1
2001 3
2002 2
2003 1
```

G Problem G

Portals and Roads

You are given **N** nodes in a line segment labeled from **1** to **N**, from left to right. So node **i** and **i + 1** are adjacent (for all **1 ≤ i < N**). Node **i** has value D_i **($D_1 < D_2 < .. < D_N$)**. You need to connect the nodes in a tree structure with portals and roads. A road can be built only between adjacent nodes. If a road is built between node **i** and **i + 1 (1 ≤ i < N)** then it takes $D_{i+1} - D_i$ time to travel that road.

A portal can be built between any **2** nodes and takes **0** time to travel. You want to build exactly R roads.

Let's denote **Time[i]** = minimum time needed to reach node **i** from node **1**. You need to connect all nodes using portals and roads in a tree structure (total portals + R **= N - 1** and all nodes are connected) in such a way that the sum of **Time[i]** for all **1 ≤ i ≤ N** is minimized. All roads and portals are bidirectional.

NOTE: A tree is an undirected connected graph in which any two nodes are connected by exactly one path, i.e. there is no cycle. So a tree of **N** nodes has exactly **N - 1** edges.

Input

The first line of the input will contain an integer **T (1 ≤ T ≤ 25)** which denotes the number of test cases. Each of the test cases starts with two integers **N(1 ≤ N ≤ 5000)** and **R(1 ≤ R ≤ 300)** describing the number of nodes and roads.The next line of the input will contain **N** space-separated integers $D_1, D_2, \ldots D_N$ **($1 \le D_i \le 10^9$, $D_i < D_{i+1}$)** describing the value of i^{th} node. The sum of **N** over all test cases does not exceed **5000**.

Output

Output `-1` if it is impossible to connect the nodes in a tree structure using exactly **R** roads. Otherwise print the minimum sum of **Time[i]** for all **1 ≤ i ≤ N**.

Sample Input	Output for Sample Input
2 5 2 1 10 11 20 21 5 3 1 11 20 30 31	2 21

Explanation:

Case 1:
Build 2 roads between these pair of nodes:
1. Node 2 and 3
2. Node 4 and 5

Then build 2 portals between these pair of nodes:
1. Node 1 and 2
2. Node 2 and 4

So, Time[1] = 0
Time[2] = 0 (portal takes no time)
Time[3] = 1
Time[4] = 0
Time[5] = 1
So Output = 2

Case 2:
Build 3 roads between these pair of nodes:
1. Node 1 and 2
2. Node 3 and 4
3. Node 4 and 5

Build 1 portal between node 1 and 4
So, Time[1] = 0
Time[2] = 10
Time[3] = 10
Time[4] = 0
Time[5] = 1
So output = 21

H

Problem H

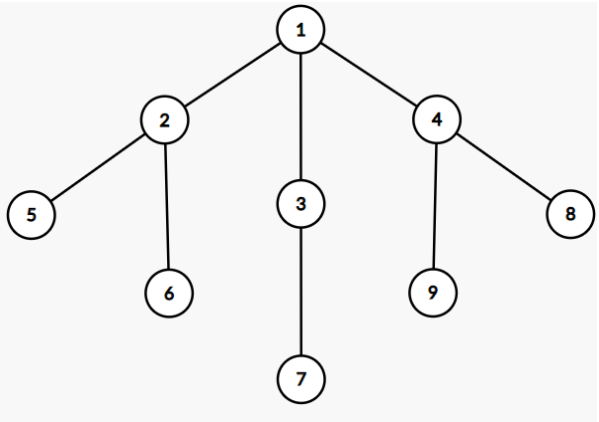
Mr. Dumb the Police in charge

Dreamland is a small, beautiful country. There are **N** cities in the country labeled from **1** to **N**. These **N** cities are connected via **N-1** bidirectional edges. The distance between any two adjacent cities is 1 and it takes 1 unit of time to go from one city to any of its adjacent cities.

Mr. Dumb is the police in charge of Dreamland. So, he gets a lot of requests from different cities to visit them in the form of **AskedHelp(T_i , C_i)**, which means the city labeled as C_i asked for help at time T_i.

Whenever Mr. Dumb is asked for help from one or more cities, he decides which one to help first based on the distance from his present location (the city where he is staying now). Amongst the requested cities, he always chooses the city with the shortest distance. If there are several such cities with the shortest distance, he chooses the city that asked for help earliest (i.e. select the city with the lowest T_i). Then he starts his journey to the chosen city.

During his journey to the selected city, he may get one or more help requests, but he will only consider helping them after reaching the city that he has already chosen (even if other cities requesting help are in the path of his journey). After reaching the destination city he will help them and will continue his journey to the next chosen city if he already has one or more requests to fulfill. Otherwise, he will wait at the current city until a new help request comes.The following example in the figure may help you understand clearly what Mr. Dumb has to do on regular basis:



The figure shows the map of the Dreamland and initially Mr. Dumb is staying in city 1.

Suppose he is asked for help 4 times as follows:
AskedHelp (3, 6)
AskedHelp (10, 7)
AskedHelp (11, 2)
AskedHelp (12, 4)
So, at time 3 Dumb will start his journey for the city labeled as 6 and will reach there at time 5 and will help them. He will wait there until the time is 10. At time 10, he will start his journey for the city labeled as 7 and reach there at 14. Then he will go to the city labeled as 2 at time 17(since the distance from city-7 to city-2 and the distance from city-7 to city-4 are equal, and city-2 asked for help earlier). Then he will go to city-4 at the time 19.

Now, you are given the map of the Dreamland and the details of the AskedHelp requests from different cities. You have to simulate and find the sequence of the cities by which Mr. Dumb will perform the help.

Input

The first line of the input will contain an integer **T (1 ≤ T ≤ 5)** which denotes the number of test cases. The first line of each test case will have an integer **N (1 ≤ N ≤ 100000)**, the number of cities in Dreamland. Each of the next N-1 lines will contain two integers U_i and V_i **(1 ≤ U_i, V_i ≤ N and U_i ≠ V_i)** which means cities labeled with U_i & V_i are connected via a bidirectional road. The next line of the test case contains an integer **M (1 ≤ M ≤ 100000)**, the number of AskedHelp requests Mr. Dumb got. Each of the next M lines will contain two integers T_i and C_i **(1 ≤ Ti ≤ 10^{15}** and **1 ≤ C_i ≤ N)** which says, city C_i asked for help at time T_i. All the requests in the input will be sorted in increasing order according to the value of T_i. It is guaranteed that no two cities will ask for help at the same time, again one city will request for help at most one time.

Output

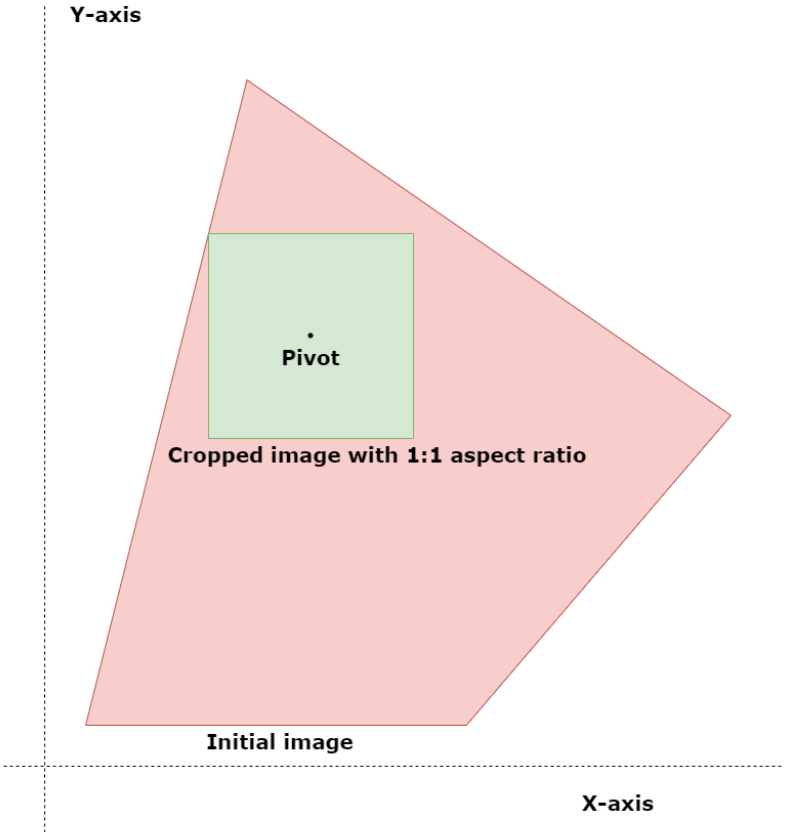
The output should start with the case number in the format "**Case X:**" where **X** is the case number and then **M** numbers should follow, separated by exactly one space which will be the order of the cities Mr. Dumb will visit. Please see the output in the sample case for more context.

Sample Input	Output for Sample Input
3 10 1 2 2 6 2 7 7 10 1 3 3 4 3 5 3 9 5 8 5 5 3 6 8 7 9 8 5 9 7 9 1 2 1 4 2 5 2 6 1 3 3 7 4 9 8 4 4 3 6 10 7 11 2 12 4 5 1 2 1 3 2 4 2 5 1 5 5	Case 1: 3 8 5 9 7 Case 2: 6 7 2 4 Case 3: 5

I Problem I

Image Cropping

Do you know how an image is cropped with a rotation and some aspect ratio? In this problem, you are given four coordinates which are the four corner points of the given image that you have to crop. You are also given a pivot point which needs to be considered to crop the image with the given aspect ratio. The pivot point needs to be at the center of the cropped image. You have to crop the image in such a way that the cropped image is a rectangle whose sides are parallel to the x-axis and the y-axis. The value of x and y coordinates of the corner points of the cropped image must be **integer**.



You have **T** test cases. For each test case, you are given four coordinates of the image, the pivot points and the required aspect ratio of the cropped image. You have to print the maximized area of the cropped image. Note, the pivot point is always **strictly inside** the initial image and the image is always a **convex quadrilateral**. Also, the area is 0 if it is not possible to have a cropped image with positive width and height.

Input

The first line will contain a single integer **T**. For each test case, there will be four lines each of which will contain the coordinates of one of the corner points of the given image ($\mathbf{P_{ix}}$, $\mathbf{P_{iy}}$). The following line of the test case will contain the coordinates of the pivot point ($\mathbf{Pivot_x}$, $\mathbf{Pivot_y}$). The value of all the coordinates are integers. The next line of the test case will contain two integers **W** and **H**, denoting the aspect ratio (the ratio of width and height - **W : H**), where **W** and **H** are coprime.

Output

For each case, print one line with "**Case <x>: <y>**", where **x** is the case number, **y** is the maximum area of the cropped image.

Constraints

- **$1 \le T \le 10^5$**
- **$-10^8 \le P_{ix}, P_{iy}, Pivot_x, Pivot_y \le 10^8$**
- **1 ≤ value of W and H for aspect ratio ≤ 10^5**

Sample Input	Output for Sample Input
2 0 0 4 0 4 4 0 4 2 2 1 1 0 0 6 0 8 4 2 4 4 2 1 1	Case 1: 16 Case 2: 16

J

Problem J

Product Quality Analyst

A factory has **N** Product Quality Analysts (PQA) working in a single line. The first PQA is standing at the start of the product line and the N^{th} PQA is standing at the end of the product line. The first PQA starts the checking of a product and takes t_1 minutes and then hands it over to the 2^{nd} PQA.
More formally, the $\mathbf{i^{th}}$ PQA takes $\mathbf{t_i}$ minutes time to check a product and then hand it over to the $(i+1)^{th}$ PQA. When, the N^{th} PQA completes his inspection for a product, we call the product a **"Verified Product"**.

In this problem, you have to calculate how much time it requires to **verify the** first **K** products.

For example, Let there be 4 PQAs working in the factory and they take 2, 3, 1, and 4 minutes time respectively to check a product. Product checking timeline these PQA is as follows:

Product		By PQA_1	By PQA_2	By PQA_3	By PQA_4
1^{st} Product	Start of checking	1	3	6	7
	End of checking	2	5	6	10
2^{nd} Product	Start of checking	3	6	9	11
	End of checking	4	8	9	14

So, the first product will be verified at the end of 10^{th} minutes, and the second product will be verified at the end of 14^{th} minutes.

Input

Input starts with an integer **T(1≤ T ≤1001)** denoting the number of test cases. Each of the test cases starts with two integers **N (1 ≤ N ≤ 20)** and **K (1 ≤ K ≤ 10^{10})**. The next line of the test case will contain **N** space-separated integers $\mathbf{t_1, t_2, \ldots t_n}$ **($1 \le t_i \le 10^8$)**.

Output

The output should be in the format "`Case X: Y`" where X is the case number and Y is the time when the K^{th} product will be verified. Please see the output in the sample case for more clarity.

Sample Input	Output for Sample Input
2 4 2 2 3 1 4 5 2 2 3 1 6 5	Case 1: 14 Case 2: 23