Brain Craft Ltd. presents BitFest 2019
powered by weDevs
Inter University Programming Contest

Hosted by CSE, KUET
Khulna, Bangladesh

7th September 2019

You get 19 Pages, 11 Problems & 300 Minutes

Platform Support:

CodeMarshal

Problem Set By:

Bangladesh Association of Problem Setters

Problem A

Stack Mania

Given two sequences of numbers **A**, **B** of length **N** and an empty stack **S**, you need to figure out if its possible to generate the sequence **B** using the stack **S** by taking the numbers from **A** sequentially. You can do one of the following operations in each iteration:

- Take the first unused number from **A** and push it to **S.**
- Pop out the top element from **S** and store it to another array **X** sequentially according to their pop order from **S.**

Iteration will stop when all the elements of **A** is used and **S** is empty. You need to figure out if its possible to generate **X** in a way that **X[i] = B[i]** where **1 ≤ i ≤ N.**

Constraints:

- Each element of **A** and **B** is unique and **1 ≤ A[i], B[i] ≤ N** for **1 ≤ i ≤ N**
- **1 ≤ N ≤ 100000**
- Sum of **N** in all test cases **≤ 1000000**

Input

First line of the input is a number **T (1 ≤ T ≤ 100)**, the number of test cases.
For each test case, first line contains an integer **N**. Next two lines contain **N** integers each which is eventually sequence **A** and **B** in that order.

Output

For each test case print a single line **“Case c: POSSIBLE|IMPOSSIBLE**” where **c** is the case number. Print **“POSSIBLE”** if its possible to generate **B** from **A**, otherwise print ”**IMPOSSIBLE**”.

Sample Input	Output for Sample Input
2 **4** **1 2 3 4** **4 3 2 1** **4** **1 2 3 4** **4 3 1 2**	**Case 1: POSSIBLE** **Case 2: IMPOSSIBLE**

Problem B

Just change it up so it doesn't look like you copied...

Alex loves spending time solving problems. He was so busy solving problems that he forgot to do his assignment. He manages to collect the assignment from his friends but realizes that he'll get caught if he doesn't change anything. So he decides to combine all of the assignments that his friends gave him into one, hoping that the teacher won't be able to tell. Now he needs your help.

Alex has **N** paragraphs that he collected from his friends. Each paragraph consists of sentences separated by a full-stop **(.)**. He wants to **create a new paragraph with exactly K sentences, by copying them from the N paragraphs.** In order to avoid being caught, he wants to make sure that **no more than 1 sentence from any single paragraph appears in the final paragraph.**

If there are multiple valid paragraphs, output the **lexicographically smallest** paragraph.
If there are no valid paragraphs output "**impossible**".

Input

The first line of input consists of **T**, which is the number of test cases. The first line of each test case contains the number **N**, which is the number of friends that have given him assignments. The next **N** lines contains one paragraph in a single line. Each paragraph consists of sentences separated by a dot **(.)**. Each sentence consists of lowercase english letters. The next line in the test case contains an integer **K**, which is the number of sentences that the final paragraph should contain.

Output

For each test case output a line containing the test case number followed by the final paragraph. If there are multiple valid paragraphs, output the lexicographically smallest one. If there are no such valid paragraphs output "**impossible**". See the sample output for clarification.

Constraints:

1 ≤ T ≤ 10

1 ≤ N, K ≤ 14

1 ≤ Number of sentences across all paragraphs in a single test case **≤ 1000**

1 ≤ Length of any sentence **≤ 1000**

Sample Input	Output for Sample Input
2 3 a.b.c c.d.e d.e.f 2 3 a.b.c c.d.e d.e.f 3	Case 1: a.d Case 2: impossible

NB: Large dataset, use faster I/O.

Problem C

Ross with another proposal

Ross proposed a girl. Of course, she didn't say yes, rather she gave him an array having **N** integers and asked him **M** queries over the array. Each query can be represented as two integers **L** and **R**.

For each query, Ross should do the following:

1. Step 1: he has to insert the numbers at index **L, L+1, L+2,……,R** of the given array into an empty multiset. Multiset is a set where an element can appear multiple times. Suppose that the size of this multiset after inserting the numbers is equal to **k**. Notice that **k** is equal to **R-L+1**.
2. Step 2: He has to generate all possible subsets of the multiset which he constructed in step **1**. For each subset he needs to **xor** the numbers of that subset. In this way, he will get 2^k values. Note that, for the empty set he will get **0**.
3. Step 3: He has to add the 2^k values which he got at step **2** and say this value modulo $10^9 + 7$ to his dream girl.

If Ross can answer all the queries correctly then she will reconsider his proposal. Can you help him to answer the queries?

Input

The first line of input contains two integers **N** and **Q**. The next line contains **N** integers, the numbers in the array. Then each of the following **Q** lines contains two integers **L** and **R**.

Constraints

- $1 \le N \le 10^5$
- $1 \le Q \le 10^5$
- $0 \le$ **Value of a number in the array** $\le 10^9$
- $1 \le L \le R \le N$

Output

For each query output an integer in a separate line, the answer for that query modulo $10^9 + 7$. Queries must be answered in the order given in the input.

Sample Input	Output for Sample Input
4 2 1 3 3 3 1 1 2 4	1 12

Problem D

And Then There Was Sum

You are given three arrays $A_1 A_2 A_3 \dots A_n$, $B_1 B_2 B_3 \dots B_m$ and $C_1 C_2 C_3 \dots C_s$. You are also given an integer **x**, **x** is a perfect power of **2**.

In one operation you can choose three elements A_i, B_j **and** C_k. An operation is considered good if the **"bitwise and"** between $(A_i + B_j + C_k)$ and **x** results in a non-zero number.

You need to calculate how many good operations are possible. Two operations are different if any of **i**, **j** or **k** is different.

Input

The First line contains an integer **T** $(1 \le T \le 10)$ denoting the number of test cases.
Each test case starts with four integers **n, m, s, x** $(1 \le n, m, s \le 10^5)$, $(x = 2^p, 0 \le p < 30)$
Second line of each test case contains **n** space separated integers $A_1 A_2 A_3 \dots A_n$ $(1 \le A_i \le 10^6)$
Third line of each test case contains **m** space separated integers $B_1 B_2 B_3 \dots B_m$ $(1 \le B_i \le 10^6)$
Fourth line of each test case contains **s** space separated integers $C_1 C_2 C_3 \dots C_s$ $(1 \le C_i \le 10^6)$

It is guaranteed that $(\max(F_A(10^5), 1) * \max(F_B(10^5), 1) * \max(F_C(10^5), 1)) \le 10^5$, Here
$F_A(y)$ denotes the number of elements present in the array **A** which are greater than **y**.
$F_B(y)$ denotes the number of elements present in the array **B** which are greater than **y**.
$F_C(y)$ denotes the number of elements present in the array **C** which are greater than **y**.

Output

For each test case, print the number of good operations in a new line. Please see the sample for details.

Sample Input	Output for Sample Input
2 5 5 5 1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 5 5 5 4 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5	63 54

NB: Large Data Set, use faster I/O.

Problem E

City Fire

Alice has **n** cities in a row. The cities are numbered from **1** to **n** from left to right. Initially none of the cities have any fire in them. At time **0**, Alice gives fire to city **1**. After a city gets fire for the first time, it keeps the fire alive forever.

Alice also has two arrays **L** and **R -** each of size **n.** The values **L[i]** and **R[i]** means that If city **i** gets fire for the first time at time **t**, then for all **x** in range **L[i] ≤ x ≤ R[i]**, city **i** throws fire to city **x** at time **t+x-L[i]+1**. If the city getting fire already has an existing fire, then nothing happens. But if it does not have any fire yet, then it catches fire for the first time and it starts to throw fire to other cities.

Now Alice asks you the times each city gets fire for the first time.

Input

The first line of the input contains a number **T (1 ≤ T ≤ 5),** denoting the number of test cases. Then description of **T** test cases follows.

First line of a test case contains an integer **n (1 ≤ n ≤ 2 * 10^5)**, indicating the number of cities. Each of the following **n** lines contains 2 space separated integers, **i'th** of which indicates values **L[i]** and **R[i] (1 ≤ L[i] ≤ R[i] ≤ n).**

Output

For each case print the case number in the first line. Then output **n** space separated numbers in the second line. **i'th** of these numbers indicates the first time city **i** catches fire. If a city never catches fire, print **-1** for it. Follow the sample output for more clarity.

Sample Input	Output for Sample Input
2 **5** **3 5** **1 2** **5 5** **1 1** **4 5** **5** **2 5** **4 5** **5 5** **1 5** **5 5**	**Case 1:** **0 -1 1 2 2** **Case 2:** **0 1 2 2 3**

Explanation of the first sample

City **1** gets fire from Alice at time **0**. It throws fire to city **3** at time **1**, city **4** at time **2** and city **5** at time **3**.

City **3** gets fire for the first time at time **1**. It throws fire to city **5** at time **2**.

City **4** gets fire for the first time at time **2**. It throws fire to city **1** at time **3**.

City **5** gets fire for the first time at time **2**. It throws fire to city **4** at time **3** and city **5** at time **4**.

City **2** never gets any fire in the process.

NB: Large Data Set, use faster I/O.

Problem F

Dumb Old Door

Alice and Bob study in Hogard School of Braincraft and Legendary. There are **N** other students in their class. Alice and Bob hangout together and they make pranks with other students all the time.

You are the famous Hairy Putter who survived the killshot from GoalThemBot. Today you just admitted into Alice and Bob's class. So, basically now there are **N+3** students in your class. **Alice**, **Bob**, **You** and **N** other students.

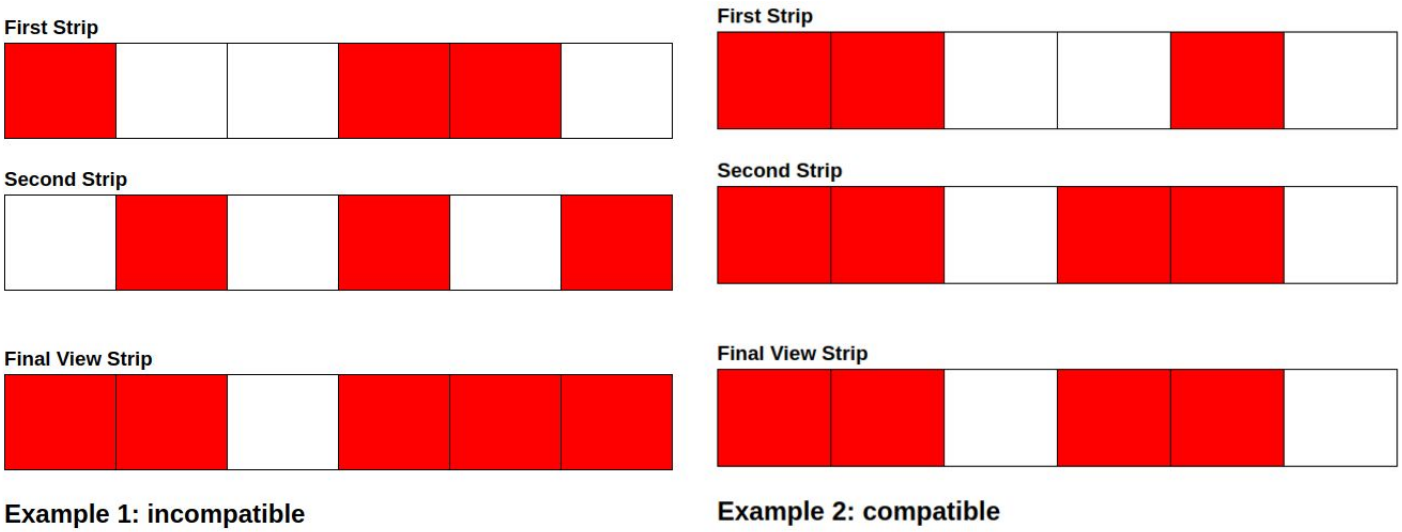
Alice and Bob decide that they must do a prank with you as you are the newest student and are very famous indeed. Alice has an **1xN** array and Bob has an **1xN** array too. Every cell of these arrays are transparent. In other words, you can see through each cell. Alice and Bob merged their arrays and basically that becomes an **1x2N** array. As all the cell is transparent, it doesn't matter how they merged those array. You can think it to be an **1x2N** array for simplicity. Let's call this merged array as "**strip**".

Alice and Bob generated infinite strips and gave them to you. They told you that you can take any strip and color some (may be none, may be all) cells with red color. You can do this with as many strips as you want. The color pattern of different strips can be the same or different too. Now they gave you a box and told you to put some strips into it. But there is a catch. If there is more than one strip in that box then any pair of strips from that box should be **incompatible**. Two strips are **compatible** to each other if you put one strip over another and the combined strip looks the same as one of the two strips.

Formally, If you put one strip over another the

- The top view of those strips will look like a single strip.
- Leftmost cell of both the strips overlap with each other, second leftmost cell of both the strips overlap with each other and so on. Of course finally the rightmost cell of both the strips overlap with each other.
- When two cells overlap, red color always dominates. If both of the cells are transparent the final view of that cell will be transparent. Otherwise the final view will be red as it dominates over transparency.

Say, one strip is **RTTRRT**. **R** denotes a red colored cell and **T** denotes a transparent cell. In this example, this is a **1x6** array. Let's say another strip is **TRTRTR**. If you put one over another (doesn't matter the order), the combined strip will look like **RRTRRR**. None of the two strips matches with this strip. So, these two strips are **incompatible**. Here is another example: If a strip is **RRTTRT** and another one is **RRTRRT**, the combined strip will be **RRTRRT** which matches with the later strip. So, these two strips are **compatible**.



Example 1: incompatible **Example 2: compatible**

Lets say, you put **X** strips in that box. Of Course no pair should be compatible. As Alice and Bob are not that bad, they told you that they will give you the same amount of dragon eggs. So, you are getting **X** dragon eggs. Not a bad deal, right? But they told you to distribute the eggs equally among the students. But they are not taking any as they already have infinite of them. So, basically you will distribute **X** eggs equally among **you** and **N** other students. Of course you want to maximize your share. Lets call your maximum possible share **S(N)**, if there are **N** students other than Alice, Bob and You in your class. You are wondering what would be **S(1), S(2), S(3), ..., S(N-1), S(N)**. Now, your task is to print the sum of all the maximum possible shares. Technically, you have to print **S(1) + S(2) + S(3) + ... + S(N-1) + S(N)**. As this number can be very big, you have to print it modulo **1000000007**.

Input

Input starts with an integer, **T**, denoting the number of test cases. For each test case there will be a line containing a single integer, **N**, denoting the number of students in your class other than Alice, Bob and You.

Output

For each test case, print a line in this format **Case X: Y**. Here, **X** denotes the case number and **Y** denotes the sum of the maximum possible shares.

Constraints

1 ≤ T ≤ 10000
1 ≤ N ≤ 50000000

Sample Input	Output for Sample Input
3 **1** **2** **9**	**Case 1: 1** **Case 2: 3** **Case 3: 6917**

Problem G

Shortest Path

Green Iceland is a dreamland. The inhabitants of that iceland are living their lives with peace, joy and happiness. Green Iceland is divided into several kingdoms. Each kingdom is divided into several cities. The cities are numbered from **1** to **N**. Cities are connected via bidirectional bridges. A bridge connects two adjacent cities. Each bridge has a bridge toll. There is at most one bridge between any two city. A particular city has at most two adjacent cities.

In this problem you will be given two cities **x, y** and you have to figure out the minimum amount of money you must pay to go to **x** from **y**.

Input

The first line contains a single integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. Each of the test cases starts with two integers **N (1 ≤ N ≤ 30000)** and **E (0 ≤ E ≤ 30000)** denoting the number of cities and bridges in Iceland. Next **E** lines contains three integers **u**, **v (1 ≤ u, v ≤ N** and **u ≠ v)** and **w (1 ≤ w ≤ 10000)** denoting that there is a bridge connecting city **u** and **v** and the toll of the bridge is **w** taka.

Next line contains an integer **Q (1 ≤ Q ≤ 50000)** denoting the number of queries you have to perform. Next **Q** queries contains two integers **x** and **y (1 ≤ x, y ≤ N).**

Output

For each case, print the case number in a single line. Then for each query print the minimum amount of money you must pay to go to **x** from **y** in a single line. If it is not possible to go to **x** from **y** print **-1** instead. See the sample I/O for more clarity.

Sample Input	Output for Sample Input
1 **4 3** **1 3 4** **2 3 3** **1 4 1** **4** **2 1** **2 3** **1 3** **2 4**	**Case 1:** **7** **3** **4** **8**

NB: Large Data Set, use faster I/O.

Problem H

Tom And Jerry Went To An Auction

Tom and Jerry are going to take part in an auction for an antique item. It's a hidden envelope auction where each participant has to submit an envelope writing down the price they are willing to pay for the item. The highest bidder wins the auction. Also none can legally know what any other person has bidden before the result is announced.

Tom has decided that he is either going to bid $\mathbf{L_T}$ dollars or $\mathbf{H_T}$ dollars for the item. Similarly Jerry has decided that he is either going to bid $\mathbf{L_J}$ dollars or $\mathbf{H_J}$ dollars for the item. We can assume that there is none except Tom and Jerry who are taking part in the auction.

You will be given the pleasure value $\mathbf{P_T}$, displeasure value $\mathbf{D_T}$ for Tom and similarly $\mathbf{P_J}$, $\mathbf{D_J}$ for Jerry. Using these values, the **satisfaction** values of Tom and Jerry can be calculated like this:

- If Tom wins the auction, his satisfaction value will be **(P_T - Tom's bid + Jerry's bid)**.
- If Tom loses the auction, his satisfaction value will be **(Jerry's Bid - Tom's Bid - D_T)**.
- Similarly, if Jerry wins the auction, his satisfaction value will be **(P_J - Jerry's bid + Tom's bid)**.
- If Jerry loses the auction, his satisfaction value will be **(Tom's Bid - Jerry's bid - D_J)**.

Both Tom and Jerry know about each other's pleasure, displeasure and values of possible bids. **Tom and Jerry both want to maximize their satisfaction value.** But the problem is, Tom thinks that Jerry can spy on him and may get to know what Tom is going to bid before the day of the auction. Jerry also thinks that Tom might do the same. So Tom makes a plan that he is not going to decide which bid he is going to put. Rather he would create a probability distribution for his two bids. For example, he can assign the probability **0.7** to the bid of $\mathbf{L_T}$ dollars and **0.3** to the bid of $\mathbf{H_T}$ dollars. Then he might roll a **10** sided dice just before the auction. If the outcome is less than or equal to **7**, he would bid $\mathbf{L_T}$, otherwise he would bid $\mathbf{H_T}$. Getting to know of this, Jerry makes a similar plan. Tom and Jerry can assign any real number between **0** and **1** to the probability of any of their bids, but the sum of probabilities assigned to the two bids of Tom have to be **1** and the same has to hold for Jerry too.

Now your task is to find a probability distribution for each of them which makes sure that none of them would want to change their probability distribution provided that he knew the other's probability distribution. **One would want to change his probability distribution if and only if he can increase his expected satisfaction value by changing the distribution considering the probability distribution of the other remains the same. It is guaranteed that at least one solution exists.**

Formally, you have to assign a value **x** to the probability of Tom bidding $\mathbf{L_T}$ and a value **y** to the probability of Jerry bidding $\mathbf{L_J}$ such that the following two conditions would hold:

i) Let S_T and S_J be the expected satisfaction values of Tom and Jerry respectively if Tom bids L_T with probability x and Jerry bids L_J with probability y. Then considering Jerry does not change his

probability distribution, S_T should be the maximum expected satisfaction value that Tom can gain for any probability distribution he can assign to his bids.

ii) Similarly, considering Tom does not change his probability distribution, S_J should be the maximum satisfaction value that Jerry can gain for any probability distribution he can assign to his bids.

Input

In the first line, the number of test cases, **T** is given. For each case, there will be two lines. The first line will contain four integers which signify L_T, H_T, P_T and D_T respectively. The next line will contain the integers signifying L_J, H_J, P_J and D_J.

Constraints

- $1 \leq T \leq 10^5$
- **All values(except for the number of test cases) in the input will be between 1 and 500**
- L_T, H_T, L_J **and** H_J **will all be different.**

Output

For each case, print one line stating the case number. Then print two fractions signifying the probability that Tom should assign to the bid L_T and the probability that Jerry should assign to the bid L_J. If there are multiple answers, you can output any answer. Your output will be considered correct if there is no other probability distribution for Tom or Jerry where they can get a higher satisfaction value considering the other keeps his distribution unchanged. The fractions should be printed in the format: ***numerator/denominator.*** Check the sample output for the exact format for printing the fractions.

You must always print both the numerator and the denominator. The fractions should be in their reduced forms i.e. the numerator and the denominator must be coprime. The value of the numerator and the denominator must be between 0 and 2000. It is guaranteed that a solution exists within the given bounds on the numerator and the denominator. The only non-negative integer coprime with 0 is 1. The denominator can never be 0.

Sample Input

Output for Sample Input

Sample Input	Output for Sample Input
1 **100 200 190 50** **120 170 130 60**	**Case 1: 0/1 1/1**

Explanation

For the sample case, Tom should always bid H_T and Jerry should always bid L_J. Then Tom will win the auction. Tom and Jerry's satisfaction value would be **110** and **20** respectively. Tom cannot increase his satisfaction by changing his decision to bid H_T always if Jerry always bids L_J. Similar argument can be given for Jerry too.

***Please use faster i/o since dataset is huge.**

Problem I

Blind Escape II

Illidan Stormage, the blind Warcraft hero is thrown in a maze for punishment. Being a blind hero, he cannot see anything but he can sense which direction is North, East, South and West.

Assume that the maze is laid out on a grid, and each grid location is either blocked or free or contains a demon. He has four possible moves: **North, East, South,** and **West**. Illidan starts from a free location at time **t=0**, and his goal is to kill all the demons within time **T.** Illidan kills a demon immediately (takes **0 unit** of time) whenever he lands on a grid location containing a demon, however making a move to an adjacent cell takes exactly **1 unit** of time. If Illidan attempts to move out of the grid or attempts to move to an obstacle, he will stay in his current location but it will still cost him **1 unit** of time. When Illidan kills the last demon, he is immediately freed from the maze but if he fails to kill all the demons within time **T**, he will be trapped in the maze for eternity.

Keep in mind that Illidan is blind, so he may get confused while making a move. To avoid all this chaos, he decided to follow a rather simple strategy. Before he starts his journey to hunt and kill all the demons, he selects four real numbers: $\mathbf{P_N}$, $\mathbf{P_E}$, $\mathbf{P_S}$ **and** $\mathbf{P_W}$ denoting the probability to move North, East, South and West respectively. No matter where he is, he will always attempt to move North with probability $\mathbf{P_N}$, East with probability $\mathbf{P_E}$, South with probability $\mathbf{P_S}$, and West with probability $\mathbf{P_W}$.

Given the description of the maze, the location of all the demons, Illidan's starting position and the probabilities of each move, calculate the probability that Illidan will be able to kill all the demons before the time runs out.

Assume the grid will be of dimensions **N** x **M** and will contain **K** demons.

Input

The first line of the input contains an integer **Q**, denoting the number of test cases. Then the description of **Q** test cases follow. The first line of every test case will start with a blank line.

The second line contains three space separated integers, **N**, **M** and **T,** denoting the number of rows, the number of columns of the grid and the time **T** by which Illidan needs to kill all the demons. The third line contains four space separated real numbers $\mathbf{P_N}$, $\mathbf{P_E}$, $\mathbf{P_S}$ **and** $\mathbf{P_W}$,in that order. All these numbers will contain at most **2 digits** after the decimal point.

Each of the next **N** lines will contain exactly **M** characters on each line describing the grid. The grid will only consist of the characters **'.', '@', '#' and '*'**.

'.' indicates a free cell
'@' indicates Illidan's starting position which is also a free cell and will occur exactly once
'**#**' denotes an obstacle
'*' denotes the location of a demon, there can be at most one demon in a cell

Constraints

- **$1 \le Q \le 5$**
- **$1 \le N, M \le 8$**
- **$1 \le T \le 16777216$**
- **$1 \le K \le 3$**
- **$P_N + P_E + P_S + P_W = 1$**

Output

For each test case, print the probability of Illidan escaping the maze in a single line. Your answer will be considered correct if the **absolute or relative error is less than 10^{-6}**. Formally, assume that your answer is defined as **A**, and the answer of the jury is defined as **B**. The checker program will consider your answer correct, iff **$abs(A - B) / max(1, B) < 10^{-6}$**.

Sample Input

```
3

1 2 3
0 0.5 0 0.5
@*

1 8 64
0 0.5 0 0.5
@......*

8 8 10000000
0.05 0.25 0.45 0.25
##....#.
.##.....
.*#.##..
.##..*#.
...###..
.#.#....
.#.#.#..
@#.*.#..
```

```
0.875000000000
0.693211582717
0.157323987423
```

Problem J

Nico's Birthday Gift

Eco and Meco are planning to gift something to their best friend Nico in his next birthday. Eco has an array **A** and Meco has another array **B.** Both of the arrays have the same length **N** . They have decided to go with a single gift which is basically a new array using any two subarrays from **A** and **B**.

After doing some extensive research, Eco and Meco have come up with the following algorithm to create the new array **C** using their two arrays **A** and **B**.

1. Remove at most **K** elements from the array **A**.
2. Remove at most **K** elements from the array **B**.
3. After removing the elements from the array **A** and **B**, the removed cells get vanished and all the remaining cells get merged together in both **A** and **B**.
4. Select any subarray **S1** from the updated array **A**.
5. Select any subarray **S2** from the updated array **B**.
6. Now you can create the array **C** by appending the subarray **S2** at the end of the subarray **S1**.
7. **C** is a valid array if all the elements of **C** are in non-decreasing order.

Now you have to find the maximum length of a valid array **C** that Eco and Meco can create following the above algorithm.

Note: A subarray **is either empty** or is a contiguous segment of an array **A** of length **N**, that can start at any index **u** and ends at any index **v** where **1 ≤ u ≤ v ≤ N**.

Input

The first line of the input contains a single integer **T**, which denotes the number of test cases. The first line of each test case contains two integers **N** and **K**. The next two lines of a test case contains the array elements of **A** and **B**.

Constraints

- **1 ≤ T ≤ 10**
- **1 ≤ N ≤ 50000**
- **0 ≤ K ≤ 10**
- **1 ≤ A_i , B_i ≤ 100000**

Output

For each test case, print a single line in the format "**Case X: R**" without the quotes where **R** denotes the maximum length of a valid array **C**.

Sample Input

Output for Sample Input

2 **5 2** **2 5 8 6 10** **12 3 15 7 18** **10 3** **2 8 18 12 25 32 19 15 18 20** **15 22 25 8 12 35 30 55 60 70**	**Case 1: 7** **Case 2: 10**

Explanation of Sample Test Case 1:

We can remove **6** from the array **A**. So the new **A** will look like this: **[2, 5, 8, 10]**.
Also we can remove **3** and **7** from the array **B**. So the new **B** will look like this: **[12, 15, 18]**.

Now we can choose **S1 = [2, 5, 8, 10]** from the array **A** and **S2 = [12, 15, 18]** from the array **B** and then append **S2** at the end of **S1** to make a new array **C**.

So **C** will become **[2, 5, 8, 10, 12, 15, 18]**. **C** is a valid array because all the elements here are in no-decreasing order.

There's no other way to maximize the length of **C** any more.

Problem K

Super Region Final is coming!

Everyone (of competitive programming community) knows that ICPC Dhaka Regional is knocking at the door for us. But, this may not be the only ICPC event for contestants of Bangladesh before World Finals this season. There is a chance we may see another contest titled "**ICPC Asia West Final Contest**"!

ICPC has been running its operations distributed among eight super regions namely: *North America, Latin America, Europe, Northern Eurasia, Africa and Arab, Asia East, Asia West* and *Asia Pacific*. While some of the other super regions had final contests before, this is the first time the ICPC officials have agreed to have a final contest for *Asia West*, the super region we belong to. Some discussed (but not finalized) rules of super region final contests are as follows:

"*A certain number of top performing teams from each regional contests under a super region will be invited to participate in the super region final.*" - *Asia West* comprises of eight regions including Dhaka. So a fixed number of top performing teams from each of these eight regional contests may get the chance to participate in the *Asia West Final*.

"*A team can only participate in the regional contest that its institution geographically belongs to and the respective super region final in a single year.*" - Bummer if this comes true. Teams from one region will not be able to attend other regional contests then. That means teams from Bangladesh can only attend Dhaka Regional contest.

"*Performance in super region final can award a slot in the next World Finals*" - Now we are talking business! Every super region is allocated with a specific number of team slots for World Finals. In *Asia West*, these slots are right now distributed among the eight regions under it. But, this will not be the case if *Asia West Final* is introduced. Some slots of *Asia West* will then be reserved for *Asia West Final* before slot distribution among all regions. If we look from a different perspective, World Finals slots reserved for a region can now be less than previous years. Whereas, if a team loses a close fight in a regional contest and fails to qualify for the World Finals, it can get another chance in the super regional final.

Everyone involved with *ICPC Asia West* operations is busy planning for this super region final. Lots of questions have appeared too - how many teams from each regionals to invite in the *Asia West Final* contest, where to hold the contest, who will be in charge of problem setting and judging, how will the judging team be formed and most importantly how many slots for each region and the final. ICPC Foundation will very soon declare the total number of slots allocated for *Asia West* super region and how to distribute this among the eight belonging regions including the final. Now we need to figure out how many teams from Bangladesh can qualify for the next ICPC World Finals, a number we always want to get larger.

*** Disclaimer: The story presented in this problem is a work of fiction and do not have any connection with reality.*

Input

First line of input contains an integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. Each of the next **T** lines contain exactly **nine** integers. First one of those, S_{AW} **(9 ≤** S_{AW} **≤ 20)** denotes the number of World Finals slots for *Asia West* super region. Second one, S_D **(1 ≤** S_D **<** S_{AW}**)** denotes the number of slots for Dhaka Region. Each of the next **seven** integers, S_i **(1 ≤** S_i **<** S_{AW}**)** denotes slots for each of the other regions under *Asia West*. You can assume, $S_D + \Sigma S_i < S_{AW}$ and the remaining slots will be reserved for the *Asia West Final* contest. You can also assume that enough number of teams from each of these regions will be selected for *Asia West Final* contest and it will not have any effect on the number of possible teams from Bangladesh to qualify in World Finals.

Output

For each test case, print a single line in the format "**Case X: D**" where **X** is the case number and **D** is the maximum number of teams from Bangladesh that can qualify for next ICPC World Finals according to the above rules.

Sample Input	Output for Sample Input
1 13 1 1 1 1 1 1 1	Case 1: 6

*** Disclaimer: The story presented in this problem is a work of fiction and do not have any connection with reality.*