A Directed Graph

Input: Standard Input
Output: Standard Output

Some people often ask what is the real-life usage of algorithm. There are tons of such usages and one popped up in the following real life scenario.

One day Alpha, Beta and Gamma went for dinner. When the bill came, we saw Alpha is charged with 23 takas, Beta with 21 takas and Gamma with 24 takas. So, Alpha gave 20 takas to Beta and Beta does not mind paying 3 takas out of his pocket for Alpha. Other than these 3 takas, Beta will pay for himself and also for Gamma. So, Beta gave 100 takas to Gamma and asked Gamma to pay to the restaurant boy. However, after a while, Beta said- "Hey Alpha, why don't you pay for Gamma today. And for my meal, here is the 20 takas, I guess you won't mind paying the rest." Alpha agreed to the proposal (and paid the restaurant) and Beta took the 100 takas back from Gamma.

Now the question is, after all these transactions how much money was spent by each one of Alpha, Beta and Gamma.

I guess you can easily solve this problem by yourself. But some people may find it difficult to solve such puzzles. In such case, it may be convenient to draw three nodes for Alpha, Beta and Gamma. And draw directed edges between nodes representing money transaction. Maybe you can add another node for the restaurant to complete the picture. So, if X pays Z amount of money to Y, we will draw an edge from X to Y with cost Z. And thus, the total amount of money paid by X will be the sum of costs of outgoing edges minus sum of costs of incoming edges. Nice, right?

Input

There will be no input to this problem.

Output

Output three numbers in a line. The amount of money paid by Alpha, Beta and Gamma from their pocket. If you think Alpha, Beta and Gamma paid 10, 20, 30 takas respectively out of their pocket then the output will look like the sample output.

Sample Input	Output for Sample Input
	`10 20 30`

Hint: Code snippet in C/C++ to print the above sample output

```
#include<stdio.h>
int main(){
        int alpha = 10, beta = 20, gamma = 30;
        printf("%d %d %d\n", alpha, beta, gamma);
        return 0;
}
```

B Wrong Solution

Input: Standard Input
Output: Standard Output

Given a list of stock values, where the i-th element is the value of the stock on the i-th day. Find the maximum money you can earn by buying and selling stocks, given that you can buy or sell (not both) only one stock on a particular day. However, you can have multiple stocks with you at any point of time. You have to buy a stock before you can sell it.

Example:

stock_prices[] = {2,4,6,8,10},Optimal solution would be { (1,5), (2,4) }(1-based indexing), which means :

1) buy a stock on day 1 and sell that on day 5.
2) buy a stock on day 2, and sell it on day 4.

This way you can earn: 10 + 8 – 2 - 4 = 12.

For this problem, someone has submitted the following code. The idea of the code is given briefly (initially all the stocks are unmarked):

1. Find the day **X** where the stock value on day **X** is the maximum among all the days not yet marked. Choose the rightmost one in case of tie.

2. Find the day **Y** where **Y** is before **X (Y < X)** and stock value on day **Y** is the minimum among all the stocks not yet marked. Choose the stock from leftmost one in case of tie.

3. Add **stock_prices[X] – stock_prices[Y]** to profit. Mark day **X** and **Y**.

4. Repeat step 1 – 3, as long as you can find both **X** and **Y**.

```
#include<bits/stdc++.h>
using namespace std;
#define MX 1002
int stock_prices[MX], nDay, marked[MX];
int getMaximum(int start, int end){
   int ret = -1, mx = 0;
   for(int i = start; i <= end; i++){
       if(!marked[i] && mx <= stock_prices[i]){
           mx = stock_prices[i];
           ret = i;
       }
   }
   return ret;
}
int getMinimum(int start, int end){
   int ret = -1, mn = 1000000000;
   for(int i = start; i ≤ end; i++){
       if(!marked[i] && mn > stock_prices[i]){
```

```
            mn = stock_prices[i];
            ret = i;
        }
    }
    return ret;
}
int getProfit(int start, int end){
    int profit = 0, i;
    for(i = start; i <= end; i++) marked[i] = 0;
    while(1){

        int X = getMaximum(start, end);

        int Y = getMinimum(start, X - 1);
        if(X==-1 || Y==-1)break;
        profit += stock_prices[X] - stock_prices[Y];
        marked[X] = marked[Y] = 1;
    }
    return profit;
}
int main(){
    scanf("%d", &nDay);
    for(int i = 1; i ≤ nDay; i++)scanf("%d", &stock_prices[i]);
    int profit = getProfit(1, nDay);
    printf("%d\n", profit);
    return 0;
}
```

But this code is actually wrong for some test cases. Find one such case. You need to write a program which will generate the case and print it as output. A sample program is given here which will produce the output of the sample output. You need to submit your program.

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int nDay = 5;
    printf("%d\n", nDay);
    printf("2 4 6 8 10\n");
    return 0;

}
```

Input

There is no input.

Output

Write a program that will print the case in the following format.
First line: Number of days, **nDay**
Second line: **nDay** numbers: **stock_prices**, which indicates the stock prices from day 1 to **nDay**.

In your test case, you need to maintain each of the following constraints:

1. **100 ≤ nDay ≤ 200**
2. **0 < stock_prices[i] ≤ 1000**
3. Optimal maximum profit for the test case must be positive.
4. Profit given by calling **getProfit(1, nDay)** must be less than the optimal maximum profit for your test case.
5. Your test case must be such that the following function should return true for your test case. You should assume that the **optimalResult(stock_prices, 1, K)** function returns the optimal maximum profit for the subarray from index 1 to k in **stock_prices**.

```
bool check(int stock_prices[], int nDay){
  int k;
  for(k = 1 ; k <= nDay - 4 ; k++ ){
      int X = getProfit(1, k);
      int Y = optimalResult(stock_prices, 1, k);
      if(X != Y) return false;
  }
  return true;
}
```

The sample output is not a correct solution to this problem. It's just to show you the format.

Sample Input

Output for Sample Input

	5 2 4 6 8 10

C Grid Construction

Input: Standard Input
Output: Standard Output

Alice have a n*m sized grid. The grid is divided into n rows and m columns. Cell(x,y) denotes the cell of the grid located in x-th row and y-th column. Alice is building wooden blocks in the grid. Each cell will contain one block and each of the blocks will contain a lowercase letter. Before starting construction, Alice has drawn a blueprint of the grid in her notebook.

Empty 2* 3 grid

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

Alice's blueprint

a	b	b
b	b	a

Alice will construct the grid in row-major order. That means, first she will complete the first row from left to right, then the second row and so on. Suppose the current block she is building must contain the letter ***c***. Now she has two choices:

* If this is the first time she is building a block which contains the letter ***c***, she will buy a brand-new block which costs ***k*** dollars.

* If there is already another block in cell ***(x, y)*** which contains the letter c, she can choose to make a copy of that block and place in the current cell ***(i, j)***. The cost for this is $|x - i| + |y - j|$ dollars. (She can choose to buy a brand new one too.)

Find the minimum cost to construct the grid. See the explanation of sample i/o for better understanding.

Input

The first line will contain ***T*** (number of test cases, $1 \leq T \leq 25$). First line of each test cases will contain three space separated integers ***n, m*** and ***k*** $(1 \leq n, m \leq 500, 0 \leq k \leq 250000)$. Each of next ***n*** lines will contain ***m*** letters denoting the blueprint of the grid. Each character of the grid is a lowercase English letter.

***Warning: Large I/O file, user faster input output functions.**

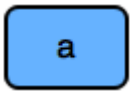
Output

Print the minimum cost to build the grid.

Sample Input	Output for Sample Input
1 2 3 2 abb bba	Case 1: 10

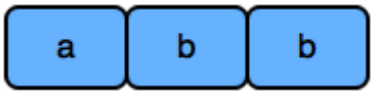
Explanation of case 1:

First she will buy a block containing ***a*** for cell (0, 0), cost = 2 dollars

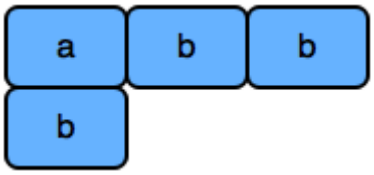


Next she will buy a block containing ***b*** for cell (0, 1), cost = 2 dollars

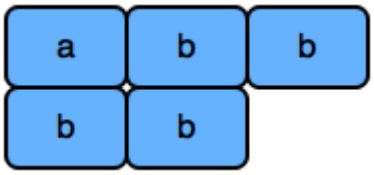
Next she will copy the block from cell (0, 1) and place it to cell (0, 2), cost = 1 dollars



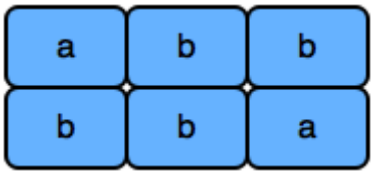
Next she will copy the block from cell (0, 1) and place it to cell (1, 0), cost = 2 dollars



Next she will copy the block from cell (0, 1) and place it to cell (1, 1), cost = 1 dollars



Next she can copy the block from cell (0, 0) and place it to cell (1, 2). But it is cheaper to buy a brand new block, cost = 2 dollars.



Total cost = 2 + 2 + 1 + 2 + 1 + 2 = 10 dollars.

D

Double Trouble

Input: Standard Input
Output: Standard Output

Camden is an evil town with **N** zones and (**N - 1**) tunnels in between the zones. The newly elected mayor of Camden ensured that every zone is reachable from all the other zones through these tunnels.

You may wonder why it is being called 'evil'. Well, people say that every zone of Camden offers some level of trouble. Depending on this level, each zone is categorized to be either Minder (less trouble) or Meer (more trouble).

And now say hello to Mr. Milo, who will be moving to Camden very soon, thanks to his new job! He was wiki-ing about Camden while he freaked out as he learned about the extreme risk of getting into trouble in the town he is relocating to. Fortunately, his office is going to be in a Minder zone, and before going to sleep, he decided that by all means he is going to live in a Minder zone as well.

But that night he had a nightmare that even though he was living in a Minder zone, on his direct way to the office from his home, he had to face, not one, but two different Meer zones!

He woke up terrified and tried to distract himself quickly, so instead of freaking out anymore, he began to think in how many ways, he can have a situation where he is living in a Minder zone which has exactly two different Meer zones on the way to the office from itself. But he is too nervous to solve it right now, can you help him?

N.B: Out of extreme anxiety after waking up, Mr. Milo forgot which Minder zone his office is situated in. So, for the problem he made up, he is considering all possible Minder zones as his possible office location.

Input

Input file contains an integer in the first line, number of test cases **T (1 ≤ T ≤ 15)**, and **T** cases follow. Each case begins with an integer **N(1 ≤ N ≤ 100000)**. The next line contains **N** space separated integers. The i'th integer is 0 if zone-i is Minder, and the i'th integer is 1 if zone-i is Meer. Then **N-1** lines follow, each line have two integers **x, y**, which denotes there is a tunnel between zone-x and zone-y.

Output

The output file contains case number, followed by the number of ways you can pick two Minder zones so that there are exactly 2 Meer zones on the direct way from one to the other.

Sample Input

```
1
5
0 1 1 0 0
1 2
2 3
3 4
4 5
```

Output for Sample Input

```
Case 1: 4
```

Explanation of Sample Test Case:
There can be four possible scenarios for zone selection as (home, office): (1, 4), (1, 5), (4, 1), (5, 1).

E Bigger Picture

Input: Standard Input
Output: Standard Output

In our country, sometimes organizers organize competition keeping a "big picture" in mind. They don't care about the fairness of a competition. They just see that "lots of people got aware." And that's a great achievement for them while a good participant's morale may get hurt by the unfairness but that gives the least amount of headache to the organizers. So, let's organize a competition ourselves with a bigger picture in mind.

There are n rooms in the competition arena. We are given the number of participants in each room. If we are to choose the first, we will look for the room with the maximum number of participants, in the case of more than one such room, we choose the room with the minimum id among the ones with maximum number of participants. After choosing the room, we randomly remove one participant from that room. Then we proceed on in the similar fashion until we rank all the participants. For example, suppose there are 3 rooms with 2, 1 and 3 participants respectively:

Number of participants	Selected room (1 based)
2, 1, 3	3 (first came from 3rd room)
2, 1, 2	1 (second from 1st room)
1, 1, 2	3 (third from 3rd room)
1, 1, 1	1 (forth from 1st room)
0, 1, 1	2 (fifth from 2nd room)
0, 0, 1	3 (sixth from 3rd room)
0, 0, 0	No participants are left

Given n and the number of participants in each of these n rooms. For the given **K**, find the room from where the **K'th** participant will be chosen from.

Input

First line of the input will be **T (≤ 20)**, number of test cases. Hence T cases will follow.

First line of the test case will contain two numbers: **n** and **m (1 ≤ n ≤ 100,000, 1 ≤ m ≤ 1000)**. Second line will contain n non-negative numbers, each not exceeding **1,000,000,000,000**. The i'th of these numbers denote number of participants in the i'th room. Third line will contain **m** numbers, each denoting a **K (1 ≤ K ≤ total number of participants**). The K's in a test case would be distinct.

Output

For each test case first print the case number, followed by m answers to the given queries. For details of the output format please check the sample input/output.

***Warning: Large I/O file, user faster input output functions.**

Sample Input

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

Output for Sample Input

```
Case 1: 1 3 3 3 1 2
Case 2: 1 2 3
Case 3: 5
```

F Counter RMQ

Input: Standard Input
Output: Standard Output

You must have heard about the Range Minimum Query (RMQ) problem. If you haven't, here is a definition:

Given an array $A[1 \ldots n]$ of n integers, the range minimum query $RMQ(l, r)$ $(1 \le l \le k \le r \le n)$ returns the value of the minimal element in the specified sub-array $A[l \ldots r]$.

Imagine a program that takes a series of RMQ queries and prints the answers. Is it possible to recover the original sequence? Yes, you are right, it's not possible for most of the cases, for different possible input arrays A, there could be the same set of answers for a given set of **RMQ** queries. Why don't we try finding one such array A?

Given n, (the length of A) and q RMQ queries and their answers in **l r RMQ(l, r)** format, print any array that produces the answer. For the purpose of this problem, we assume **A[i]** will be positive and less than or equal to 20000.

Input

The input starts with T, the number of test cases $(T \le 100)$. It follows T sets of test cases.
Each test case starts with two numbers n and q $(1 \le n \le 100, 1 \le q \le 20000)$ denoting the array length and the number of queries. That is followed by **q** lines. Each of these **q** lines consists of three integers **l, r, RMQ(l, r)** as mentioned in the statement. (**1 ≤ l ≤ r ≤ n and 1 ≤ RMQ(l, r) ≤ 20000**). You can assume that there will always be at least one valid answer consistent with the input.

***Warning: Large I/O file, user faster input output functions.**

Output

For each set of inputs, print the test case number followed by the output array, **n** integers separated by single space. For the output format, please consult the sample input/output. There can be multiple correct answers, you can print anyone.

Sample Input	Output for Sample Input
1 7 5 1 4 4 4 7 12 2 2 17 1 7 4 4 5 32	Case 1: 10 17 4 57 32 53 12

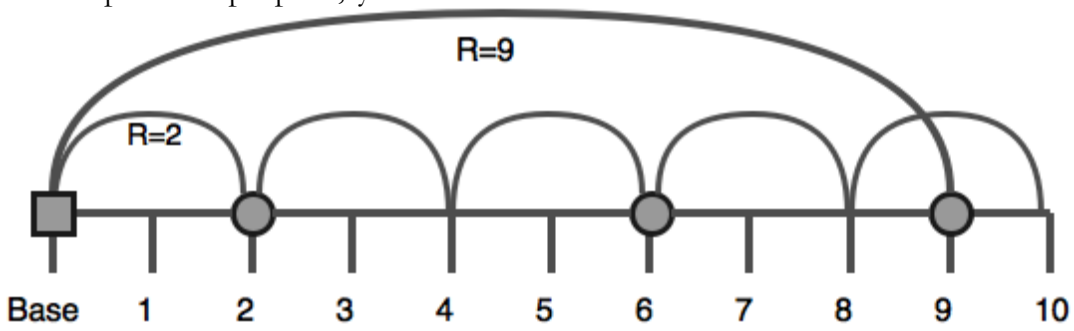
G Repeater

Input: Standard Input
Output: Standard Output

Hello soldier, welcome to the military base Ground Zero. The enemies are planning to attack our base. Right now, they are camping in the jungle near our camp for the night. We must destroy them before the sunrise.

Now there are **N** enemy vehicles in a straight line. The distance of the vehicles are d_1, d_2, d_3, d_4,,, d_n. The distances are all positive integers between 1 to **M**.

For all practical purpose, you can think of our base and all the vehicles are dots in a straight line.



Now the enemy doesn't know about our secret weapon the repeater. The repeater is a powerful weapon with which can destroy the enemy vehicles. When firing the repeater, we can choose a parameter **R** and the repeater will first destroy everything at exactly **R** distance away, then it will destroy everything in 2**R** distance, then 3**R**, 4**R** until all the multiples of **R** less than or equal **M** are destroyed. Once fired, the repeater can't be stopped. Now your job is to operate the repeater.

Now the cost of firing the repeater with parameter **R** is Floor(**M**/**R**). We want to destroy all the enemy vehicles but we also want to minimize the cost. You can fire the repeater as many times as you want.

So now if **M** is 10 and enemy vehicles are at 2, 6 and 9 distance away, we can fire the repeater with parameter R=1 and destroy them but the cost will be Floor(10/1) = 10. Better approach is firing with parameter R=2 and R=9 where the total cost will be Floor(10/2) + Floor(10/9) = 5 + 1 = 6.

So, soldier, your mission is to destroy all the enemy vehicles with minimum cost. Don't disappoint us.

Input

First line of the input is **T(≤100)**, then **T** test cases follows. Each case start with a line containing 2 positive integers **N (1 ≤ N ≤ 17)** the number of enemy vehicles and **M (1 ≤ M ≤ 1000000)**. Next line will be **N** positive integers $\mathbf{d_1}$, $\mathbf{d_2}$, $\mathbf{d_3}$ **,,,** $\mathbf{d_n}$ **(1 ≤** $\mathbf{d_i}$ **≤ M)** which are distances of enemy vehicles.

Output

For each test case print a line in "**Case I: C**" format where **I** is case number and **C** is the minimum cost for destroying all the enemy vehicles.

Sample Input	Output for Sample Input
1 3 10 2 6 9	Case 1: 6

H An Interesting Game

Input: Standard Input
Output: Standard Output

Let's play a game. You might like it.

You are given an array of **N** integers. All the numbers are distinct and between **[1...N].**

The rule of the game is simple. You will be given two integers, let's call them **L, R (1 ≤ L ≤ R ≤ N)**. You need to find out the length of the largest contiguous sub-array of the given permutation in which every value is between **L** and **R inclusive**. You will be given many queries like this for the given array. You win if you can answer all the queries perfectly and in time. Can you win this game?

Input

The first line has an integer **T** (The number of test scenarios, **1≤ T ≤ 5**). Then T test cases follow. Each test had two integers at the first line **N, Q (1 ≤ N, Q ≤ 100000)**. Then the next line contains a permutation of positive integers between **1...N**. The next **Q** lines contain pairs of integers: **L** and **R**.

***Warning: Large I/O file, user faster input output functions.**

Output

For each test case print the case number at the first line as "**Case T:**", where **T** is the test case number. Then for each test case, print **Q** lines, each line should contain the length of the largest sub-array of the array which contains all the values only from [L, R] (inclusive). See the sample input/output for more details.

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

I Gadgets of Tomishu

Input: Standard Input
Output: Standard Output

Little Tomishu likes numbers. But he likes electronic gadgets (mobile, tablet, phablet, laptop and so on) more than numbers. His parents are trying to keep him engaged with numbers. So, they asked him to write down all the numbers of n length, consisting of only 0 and 1, and without any neighboring 1s. For example, if n = 3, he is supposed to write: 000, 001, 010, 100 and 101. Of course, Tomishu will write the numbers in increasing order. For each of the numbers, his parent will reward him with one of the k types of gadgets. Find out the number of ways he can get the gadgets. Two ways will be different if he received different gadget for the i^{th} number for some i. For example, for n = 3, if there are k = 2 types of gadgets (say mobile and laptop), he can get the gadgets in $2 * 2 * 2 * 2 * 2 = 2^5 = 32$ ways.

Since the answer can be quite big for the given n and k, you will also be given **M**. You need to provide the answer modulo **M**.

However, you should know that recently Tomishu's interest shifted from electronic gadget to bookfacing.

Input

First line of the input is **T (≤ 20)**, the number of test cases. Hence **T** lines will follow. Each of these lines will contain 3 positive integers **n, k** and **M (1 ≤ n, k ≤ 100,000** and **1 ≤ M ≤ 1,000,000,000)**.

Output

For each of the cases print case number, followed by the answer. For details, please check the sample output.

Sample Input	Output for Sample Input
2 3 2 100 4 3 1000	Case 1: 32 Case 2: 561