Codeforces Round #652 (Div. 2)

A. FashionabLee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lee is going to fashionably decorate his house for a party, using some regular convex polygons...

Lee thinks a regular n-sided (convex) polygon is *beautiful* if and only if he can rotate it in such a way that at least one of its edges is parallel to the OX-axis and at least one of its edges is parallel to the OY-axis at the same time.

Recall that a regular n-sided polygon is a convex polygon with n vertices such that all the edges and angles are equal.

Now he is shopping: the market has t regular polygons. For each of them print YES if it is beautiful and NO otherwise.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of polygons in the market.

Each of the next t lines contains a single integer n_i ($3 \le n_i \le 10^9$): it means that the i-th polygon is a regular n_i-sided polygon.

Output

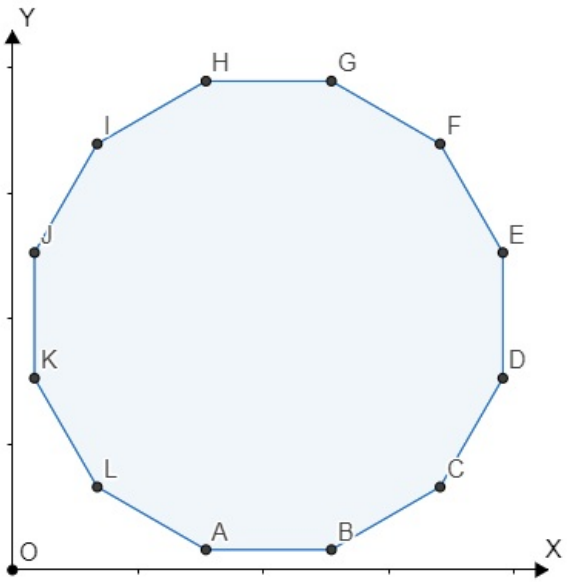
For each polygon, print YES if it's beautiful or NO otherwise (case insensitive).

Example

input
4 3 4 12 1000000000
output
NO YES YES YES

Note

In the example, there are 4 polygons in the market. It's easy to see that an equilateral triangle (a regular 3-sided polygon) is not beautiful, a square (a regular 4-sided polygon) is beautiful and a regular 12-sided polygon (is shown below) is beautiful as well.



B. AccurateLee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input

output: standard output

Lee was cleaning his house for the party when he found a messy string under the carpets. Now he'd like to make it clean accurately and in a stylish way...

The string s he found is a binary string of length n (i. e. string consists only of 0-s and 1-s).

In one move he can choose two consecutive characters s_i and s_{i+1}, and if s_i is 1 and s_{i+1} is 0, he can erase **exactly one of them** (he can choose which one to erase but he can't erase both characters simultaneously). The string shrinks after erasing.

Lee can make an arbitrary number of moves (possibly zero) and he'd like to make the string s as *clean* as possible. He thinks for two different strings x and y, the **shorter** string is cleaner, and if they are the same length, then the *lexicographically smaller* string is cleaner.

Now you should answer t test cases: for the i-th test case, print the cleanest possible string that Lee can get by doing some number of moves.

Small reminder: if we have two strings x and y of the same length then x is lexicographically smaller than y if there is a position i such that $x_1 = y_1$, $x_2 = y_2$,..., $x_{i-1} = y_{i-1}$ and $x_i < y_i$.

Input
The first line contains the integer t ($1 \le t \le 10^4$) — the number of test cases.

Next $2t$ lines contain test cases — one per two lines.

The first line of each test case contains the integer n ($1 \le n \le 10^5$) — the length of the string s.

The second line contains the binary string s. The string s is a string of length n which consists only of zeroes and ones.

It's guaranteed that sum of n over test cases doesn't exceed 10^5.

Output
Print t answers — one per test case.

The answer to the i-th test case is the cleanest string Lee can get after doing some number of moves (possibly zero).

Example

input
5 10 0001111111 4 0101 8 11001101 10 1110000000 1 1
output
0001111111 001 01 0 1

Note
In the first test case, Lee can't perform any moves.

In the second test case, Lee should erase s_2.

In the third test case, Lee can make moves, for example, in the following order: 11001101 → 1100101 → 110101 → 10101 → 1101 → 101 → 01.

C. RationalLee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lee just became Master in Codeforces, and so, he went out to buy some gifts for his friends. He bought n integers, now it's time to distribute them between his friends rationally...

Lee has n integers $a_1, a_2, \ldots, a_n$ in his backpack and he has k friends. Lee would like to distribute **all** integers in his backpack between his friends, such that the i-th friend will get exactly w_i integers and each integer will be handed over to exactly one friend.

Let's define the *happiness* of a friend as the sum of the maximum and the minimum integer he'll get.

Lee would like to make his friends as happy as possible, in other words, he'd like to maximize the sum of friends' happiness. Now he

asks you to calculate the maximum sum of friends' happiness.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Next $3t$ lines contain test cases — one per three lines.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le n$) — the number of integers Lee has and the number of Lee's friends.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the integers Lee has.

The third line contains k integers $w_1, w_2, \ldots, w_k$ ($1 \le w_i \le n$; $w_1 + w_2 + \ldots + w_k = n$) — the number of integers Lee wants to give to each friend.

It's guaranteed that the sum of n over test cases is less than or equal to $2 \cdot 10^5$.

Output
For each test case, print a single integer — the maximum sum of happiness Lee can achieve.

Example

input
3 4 2 1 13 7 17 1 3 6 2 10 10 10 10 11 11 3 3 4 4 1000000000 1000000000 1000000000 1000000000 1 1 1 1
output
48 42 8000000000

Note
In the first test case, Lee should give the greatest integer to the first friend (his happiness will be $17 + 17$) and remaining integers to the second friend (his happiness will be $13 + 1$).

In the second test case, Lee should give $\{10, 10, 11\}$ to the first friend and to the second friend, so the total happiness will be equal to $(11 + 10) + (11 + 10)$

In the third test case, Lee has four friends and four integers, it doesn't matter how he distributes the integers between his friends.

D. TediousLee

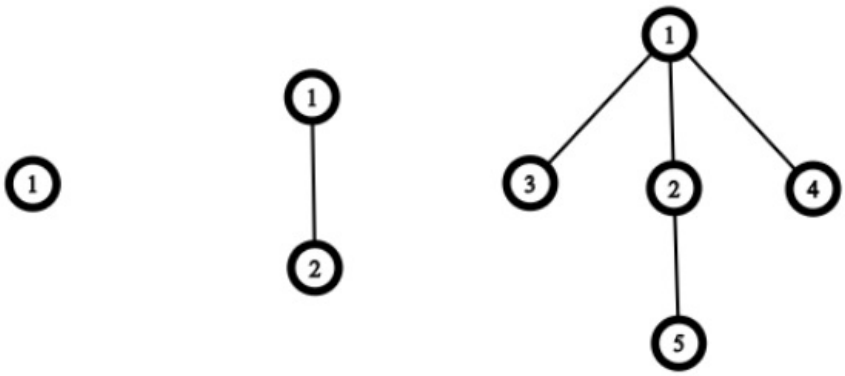
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lee tried so hard to make a good div.2 D problem to balance his recent contest, but it still doesn't feel good at all. Lee invented it so tediously slow that he managed to develop a phobia about div.2 D problem setting instead. And now he is hiding behind the bushes...

Let's define a *Rooted Dead Bush* (RDB) of level n as a rooted tree constructed as described below.

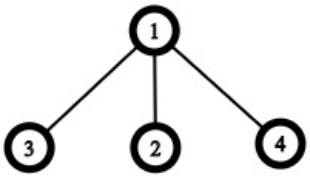
A rooted dead bush of level 1 is a single vertex. To construct an RDB of level i we, at first, construct an RDB of level $i - 1$, then for each vertex u:

- if u has no children then we will add a single child to it;
- if u has one child then we will add two children to it;
- if u has more than one child, then we will skip it.



Rooted Dead Bushes of level 1, 2 and 3.

Let's define a *claw* as a rooted tree with four vertices: one root vertex (called also as center) with three children. It looks like a claw:



The center of the claw is the vertex with label 1.

Lee has a Rooted Dead Bush of level n. Initially, all vertices of his RDB are green.

In one move, he can choose a claw in his RDB, if all vertices in the claw are *green* and all vertices of the claw are children of its center, then he colors the claw's vertices in yellow.

He'd like to know the maximum number of yellow vertices he can achieve. Since the answer might be very large, print it modulo $10^9 + 7$.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Next t lines contain test cases — one per line.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^6$) — the level of Lee's RDB.

Output

For each test case, print a single integer — the maximum number of yellow vertices Lee can make modulo $10^9 + 7$.

Example

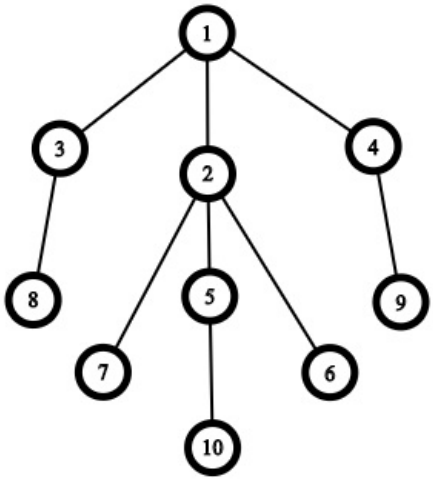
input
7 1 2 3 4 5 100 2000000
output
0 0 4 4 12 990998587 804665184

Note

It's easy to see that the answer for RDB of level 1 or 2 is 0.

The answer for RDB of level 3 is 4 since there is only one claw we can choose: $\{1, 2, 3, 4\}$.

The answer for RDB of level 4 is 4 since we can choose either single claw $\{1, 3, 2, 4\}$ or single claw $\{2, 7, 5, 6\}$. There are no other claws in the RDB of level 4 (for example, we can't choose $\{2, 1, 7, 6\}$, since 1 is not a child of center vertex 2).



Rooted Dead Bush of level 4.

E. DeadLee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lee bought some food for dinner time, but Lee's friends eat dinner in a deadly way. Lee is so scared, he doesn't want to die, at least not before seeing Online IOI 2020...

There are n different types of food and m Lee's best friends. Lee has w_i plates of the i-th type of food and each friend has two different favorite types of food: the i-th friend's favorite types of food are x_i and y_i ($x_i \neq y_i$).

Lee will start calling his friends one by one. Whoever is called will go to the kitchen and will try to eat **one plate of each of his favorite food types**. Each of the friends will go to the kitchen exactly once.

The only problem is the following: if a friend will eat at least one plate of food (in total) then he will be harmless. But if there is nothing left for him to eat (neither x_i nor y_i), he will eat Lee instead $\times\_\times$.

Lee can choose the order of friends to call, so he'd like to determine if he can survive dinner or not. Also, he'd like to know the order itself.

Input

The first line contains two integers n and m ($2 \leq n \leq 10^5$; $1 \leq m \leq 2 \cdot 10^5$) — the number of different food types and the number of Lee's friends.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \leq w_i \leq 10^6$) — the number of plates of each food type.

The i-th line of the next m lines contains two integers x_i and y_i ($1 \leq x_i, y_i \leq n$; $x_i \neq y_i$) — the favorite types of food of the i-th friend.

Output

If Lee can survive the dinner then print `ALIVE` (case insensitive), otherwise print `DEAD` (case insensitive).

Also, if he can survive the dinner, print the order Lee should call friends. If there are multiple valid orders, print any of them.

Examples

input
3 3 1 2 1 1 2 2 3 1 3
output
ALIVE 3 2 1

input
3 2 1 1 0 1 2 1 3
output
ALIVE

2 1

input
4 4 1 2 0 1 1 3 1 2 2 3 2 4
output
ALIVE 1 3 2 4

input
5 5 1 1 1 2 1 3 4 1 2 2 3 4 5 4 5
output
ALIVE 5 4 1 3 2

input
4 10 2 4 1 4 3 2 4 2 4 1 3 1 4 1 1 3 3 2 2 1 3 1 2 4
output
DEAD

Note

In the first example, any of the following orders of friends are correct : $[1, 3, 2]$, $[3, 1, 2]$, $[2, 3, 1]$, $[3, 2, 1]$.

In the second example, Lee should call the second friend first (the friend will eat a plate of food 1) and then call the first friend (the friend will eat a plate of food 2). If he calls the first friend sooner than the second one, then the first friend will eat one plate of food 1 and food 2 and there will be no food left for the second friend to eat.

F. BareLee

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lee is used to finish his stories in a stylish way, this time he barely failed it, but Ice Bear came and helped him. Lee is so grateful for it, so he decided to show Ice Bear his new game called "Critic"...

The game is a one versus one game. It has t rounds, each round has two integers s_i and e_i (which are determined and are known before the game begins, s_i and e_i may differ from round to round). The integer s_i is written on the board at the beginning of the corresponding round.

The players will take turns. Each player will erase the number on the board (let's say it was a) and will choose to write either $2 \cdot a$ or $a + 1$ instead. Whoever writes a number strictly greater than e_i loses that round and the other one wins that round.

Now Lee wants to play "Critic" against Ice Bear, for each round he has chosen the round's s_i and e_i in advance. Lee will start the first round, the loser of each round will start the next round.

The winner of the last round is the winner of the game, and the loser of the last round is the loser of the game.

Determine if Lee can be the winner independent of Ice Bear's moves or not. Also, determine if Lee can be the loser independent of Ice Bear's moves or not.

Input

The first line contains the integer t ($1 \leq t \leq 10^5$) — the number of rounds the game has.

Then t lines follow, each contains two integers s_i and e_i ($1 \leq s_i \leq e_i \leq 10^{18}$) — the i-th round's information.

The rounds are played in the same order as given in input, s_i and e_i for all rounds are known to everyone before the game starts.

Output

Print two integers.

The first one should be 1 if Lee can be the winner independent of Ice Bear's moves, and 0 otherwise.

The second one should be 1 if Lee can be the loser independent of Ice Bear's moves, and 0 otherwise.

Examples

input
3 5 8 1 4 3 10
output
1 1

input
4 1 2 2 3 3 4 4 5
output
0 0

input
1 1 1
output
0 1

input
2 1 9 4 5
output
0 0

input
2 1 2 2 8
output
1 0

input
6 216986951114298167 235031205335543871 148302405431848579 455670351549314242 506251128322958430 575521452907339082 1 768614336404564650 189336074809158272 622104412002885672 588320087414024192 662540324268197150
output
1 0

Note

Remember, whoever writes an integer greater than e_i loses.