2016 acmicpc THAILAND CENTRAL A CONTEST

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University

ACM-ICPC Thailand Central A Contest
September 11, 2016

Contest Session


ACM-ICPC 2016

Thailand Central A Contest

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING, CHULALONGKORN UNIVERSITY

SEPTEMBER 11, 2016

Memory limit (all problem):

2048MB

Time limit (seconds)

A	1
B	1
C	3
D	1
E	2
F	3
G	1
H	1
I	1
J	5
K	1
L	1

A Jumping Worm

Time Limit | **1 second**

The forest where a little worm lives has **N** trees, neatly located at positions **1** to **N**. The worm dreams to be the first ever being to place its slimy belly on the moon… Nah, not really. Well, he just wants to be at the top of the N^{th} tree. Right now, the little worm is at the base of the 1^{st} tree. He can move in one of the following two ways:

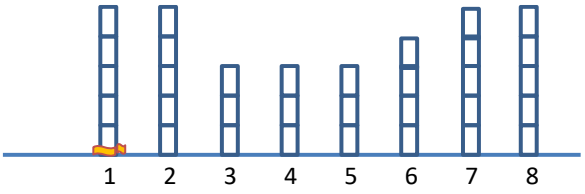
1. When he is on the i^{th} tree, he can climb up u_i units at a time. The height after any climb will never be higher than the height of the tree.
2. He can jump from the i^{th} tree to an adjacent tree (the $(i-1)^{th}$ or the $(i+1)^{th}$ tree). The worm is a good jumper, so he can jump to the same height on the nearby tree. **However, if the top of the adjacent tree is lower than the worm's current height, the worm will end up on the top of this lower tree.**

Each move takes the worm 1 second. But, after each move, the worm has to rest for 1 second. During this time, he will fall down due to gravity for d_i units if he is on the i^{th} tree. Yet he cannot go below height 0. There is an exception to this rule: when he jumps to the top of a tree, he does not have to rest; he can continue moving right after his previous move. Because trees are generally different, the u_i and d_i of different tree can be different.

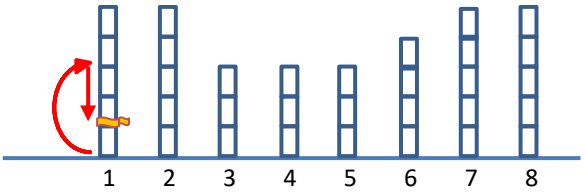
Remarks: Each of the two maneuvers requires exactly 1 second, even in the cases where the movement does not go the full length because it reaches the top or the bottom of a tree.

Example: There are 8 trees, whose heights from left to right are 5, 5, 3, 3, 3, 4, 5, and 5 units, as shown below. All the $\mathbf{u}_i$'s are 3 units per second, and the $\mathbf{d}_i$'s are 2 units per second. Figures a) to g) show one possible way for the little worm to move from the starting position, the base of the first tree, to the top of the 8th tree in **Figure f)**.

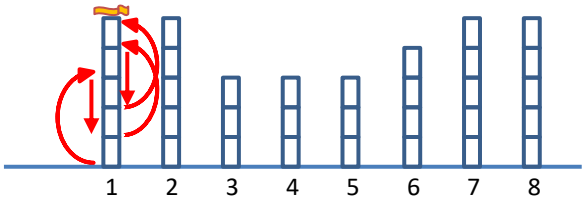
a) The little worm begins at the base of the first tree.



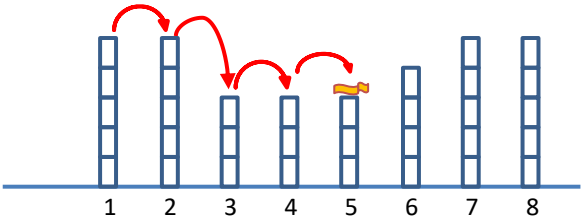
b) After the worm climbed 3 units up and fell 2 units down during the wait, he ended up at the position of the flag.



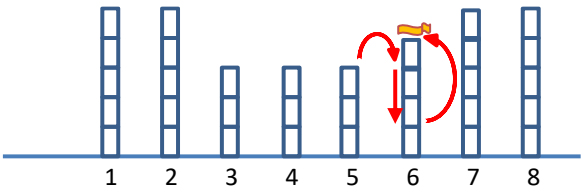
c) He must spend 5 seconds to get to the top.



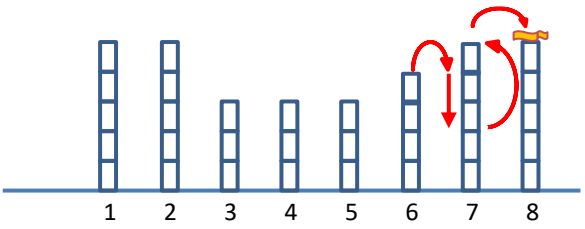
d) Then, he jumpped to the tree 2, 3, 4, 5 on the 6th, 7th, 8th, 9th seconds respectively without having to rest.



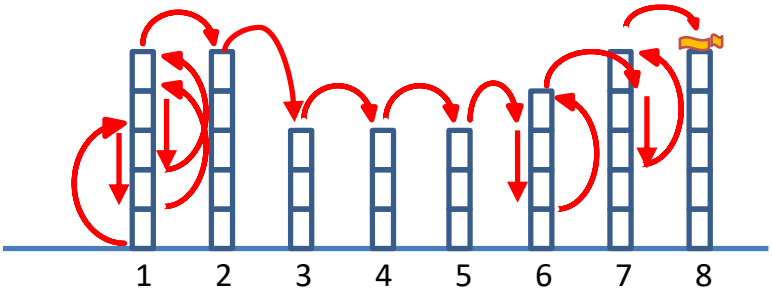
e) While on the 6th tree, the little worm was not at the top, so he must rest for 1 second. During the time, he fell down 2 units. Then, he moved to the top of the tree again.



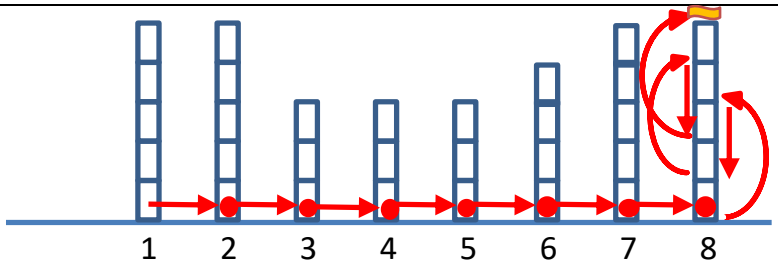
f) The little worm continued its journey to the top of the 8th tree.



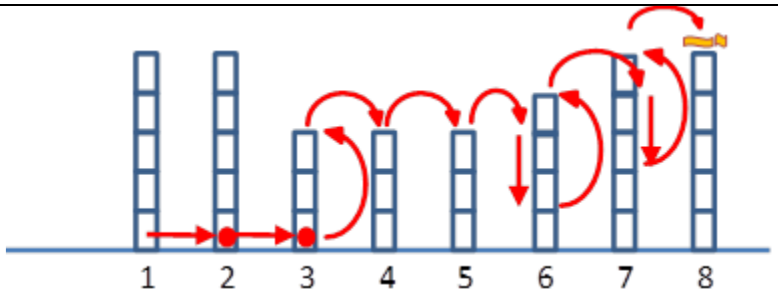
g) This figure summarized the trip of the little worm from **Figure a)** to **Figure f)**. The total time required for this trip is **16 seconds**.



Now if the worm decides to instead jump from the base of one trees to the next as shown in the figure below, and only climbs up on the last tree, he will take a total of 19 seconds. Note that since the jump is not made at the top, he must rest for 1 second (shown as dots). Also, since the worm is at the base, he cannot fall further while resting.



It turns out for this particular example, the shortest possible travel time for the little worm is 14 seconds, as shown in the figure below.



Your task

Given **N** trees, the u_i and d_i of all trees, find the shortest travel time (in seconds) from the base of the 1st tree to the top of the **N**th tree.

Input

For the first line, there is a single integer, **K**, representing the number of test cases. For each test case, there are 4 lines. ($1 \le K \le 10$)

1) The first line contains a single integer, **N**, the number of the trees ($0 < N \le 1000$).
2) The second line contains **N** numbers, h_i, where **i** goes from 1 to **N**. Note that h_i represents the height of each tree. ($1 \le h_i \le 1000$)
3) The third line contains **N** numbers, u_i, where **i** goes from 1 to **N**. Note that u_i represents the climb rate of each tree. ($1 \le u_i \le 1000$)
4) The fourth line contains **N** numbers, d_i, where **i** goes from 1 to **N**. Note that d_i represents the fall rate of each tree. ($1 \le d_i \le 1000$)

Output

For each test case, you print out on a single line the shortest travel time of the little worm. However, if it is not possible to reach the top of the **N**th tree, you must print, in capital letter, the word "**NEVER**", without the quotation.

Example

Input	Output
1 8 5 5 3 3 3 4 5 5 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2	14

3 8 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 5 5 3 3 3 4 5 5 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 8 5 5 3 3 3 4 5 5 3 3 1 1 1 1 3 1 2 1 0 4 0 4 1 2	NEVER 11 16

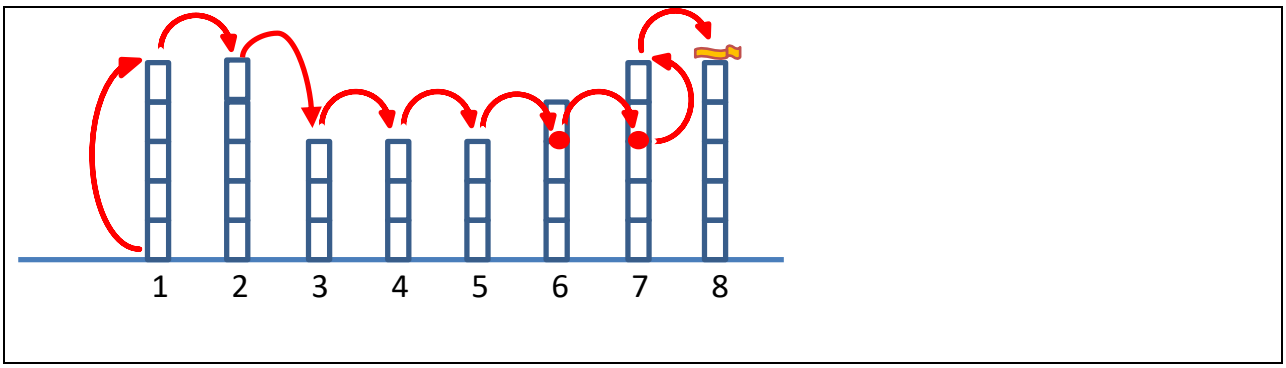
Example case description

First example cases is the case described above.

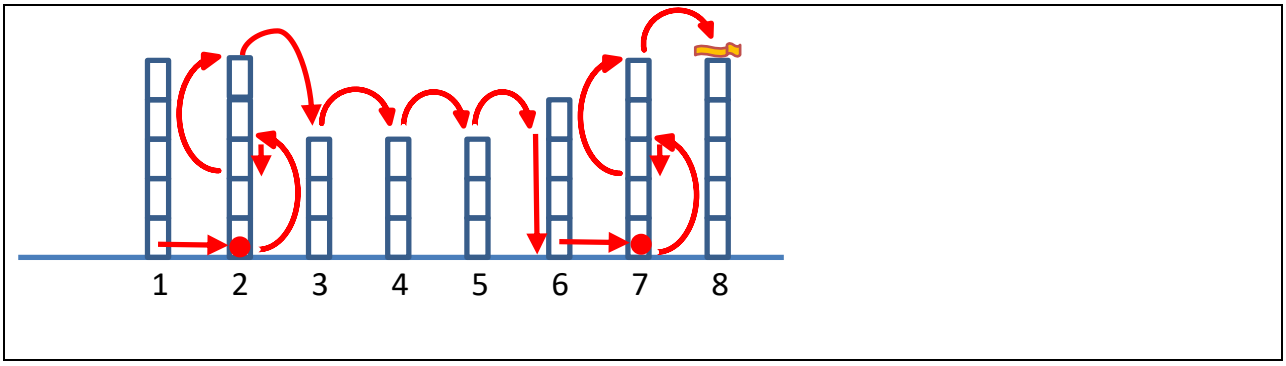
For the second example cases:

First test case: The poor little worm cannot climb at all because after each climb of 1 unit, he also fell down 1 unit.

Second test case: Even if he can climb 6 units in one second on the first tree, the tree's height is only 5 units. Hence, he ended up there at the top of the 1st tree after 1 second. After the jump to the 6th tree, the worm did not fall down because $d_i = 0$. However, he still needed to rest for 1 second (as shown in dot). The same thing happened on the next jump to the 7th tree. Finally, he made the move as shown in the figure below, using the total of 11 seconds.



Third case: The shortest travel time is 16 seconds. The worm can travel as shown in the figure below.



B

Math Contest

Time Limit | **1 second**

International Mathematical Olympiad (IMO) 2016 was held on 11th – 12th July. There are 3 problems in each day and one of them is as follows.

Problem 2. Find all positive integers n for which each cell of an $n \times n$ table can be filled with one of the letters *I*, *M* and *O* in such a way that:

- In each row and each column, one third of the entries are *I*, one third are *M* and one third are *O*; and
- In any diagonal, if the number of entries on the diagonal is a multiple of three, then one third of the entries are *I*, one third are *M*, and one third are *O*.

Note: The rows and columns of an $n \times n$ table are each labelled 1 to n in a natural order. Thus each cell corresponds to a pair of positive integers (i,j) with $1 \le i, j \le n$. For $n > 1$, the table has $4n-2$ *diagonals* of two types. A diagonal of the first type consists of all cells (i,j) for which $i+j$ is a constant, and a diagonal of the second type consists of all cells (i,j) for which $i-j$ is a constant.

The answer of this question is all positive integers which is divisible by 9 ($n = 9k$ for some positive integer k). However, you are now participating ACM-ICPC contest, not IMO, so the question is changed to "Determine whether the given positive integer x can be used as n in the given problem?"

Input

The first line contains the number of test cases **T** ($1 \le T \le 1000$)

For each test case, there will be an integer **x** ($1 \le x \le 10^{100\,000}$)

Output

For each test case, print '**YES**' if ***x*** can be used as ***n***, and '**NO**' otherwise, both without quote.

Example

Input	Output
3 1 81 999999999999999999999999999999	NO YES YES

C Weeping Fig

Time Limit | **3 seconds**

"Roots grow rapidly, invading gardens, growing under and lifting sidewalks, patios, and driveways." -- *United States Forest Service*

Because of that, it is not recommend to be planted in residential household. However, you happen to have one in your backyard, and your fig is very nice — it does not invade any part of your household at all. It just expands into empty space around the house.

Your friend are constructing a new house, and want to have a weeping fig in his backyard too. But he is scared that traditional weeping fig could destroy his home, so he wants your weeping fig (which is now known among your friends to be a "nice weeping fig.") Since the tree grows rapidly, you decided to just cut parts of your tree for your friend.

Even though just only one root is enough for your friend to have the same nice tree as you, it is not easy separating one root since it has many connections to other roots. Specifically, each root has many connections to some other roots, with no pattern. Each root connection also vary in diameter, thus the effort required for separating is not constant.

Given the number of roots, its connection and connection separation effort, please find the minimal effort required to separate your weeping fig into two entities.

Input

First line has the number of test cases, **T** ($1 \le T \le 20$)

For each test case, the first line contains two integers **N M** specifying the number of roots and number of connection ($1 \le N \le 500$, $N-1 \le M \le (N^2-N)/2$)

Following **M** lines contain three integer a_i b_i w_i ($1 \le a_i, b_i \le N$, $1 \le w_i \le 1000$) indicating that there is a root connection from root a_i to root b_i that requires effort w_i to separate. It is guaranteed that the input will form single weeping fig tree only.

Output

For each test case, output the minimal effort required on its own line.

Example

Input	Output
1 4 6 1 2 2 1 3 3 1 4 5 2 3 5 2 4 5 3 4 8	10

Explanation

You can split the node 1 from the rest, which takes 10 effort. If you instead split 12 and 34, you would have 8 + 10 = 18 effort.

D

Cocktail Shaker Sort

Time Limit | **1 second**

Cocktail Shaker Sort is a variation of bubble sort. Shaker sort, unlike bubble sort, orders the array in both directions.

Given a sequence of **N** positive integers that contains unique integers from 1 to **N**. A process of Cocktail Shaker Sort described:

1. Move number '1' to the first position by swapping one-by-one with the number next to it until it stays at the first position.
2. Move number '**N**' to the last position by swapping one-by-one with the number next to it until it stays at the last position.
3. Move number '2' to the second position by swapping one-by-one with the number next to it until it stays at the second position.
4. Move number '**N**-1' to the (**N**-1)th position by swapping one-by-one with the number next to it until it stays at the (**N**-1)th position.
5. Continue moving until the **N**th step has been completed.

For example, when **N**=6 and the initial sequence is 654321. In the first step, the number '1' is swapped 5 times and the sequence becomes 165432. The number '6' is then swapped 4 times in the second step and the sequence becomes 154326. In the third step, the number '2' is swapped 3 times and the sequence becomes 125436. In the fourth step, the number '5' is swapped 2 times and the sequence becomes 124356. Then, the number '3' is swapped 1 time and the sequence becomes 123456. Lastly, no more swapped is needed in the 6th step as the number '4' is already in the 4th position.

Your task

Write a program to count the number of swaps needed in each step of the Cocktail Shaker Sort.

Input

The first line of the input contains integer **T**, the number of test cases. ($1 \le T \le 15$) Then **T** test cases follow in the format described next.

The first line of each test case contains a positive integer **N** ($1 \le N \le 100\,000$)

The second line of each test case contains **N** unique positive integers from 1 to **N** separated by one space.

Output

The output contains **T** lines, each line shows the number of swaps needed in each step of the Cocktail Shaker Sort separated by one space.

Example

Input	Output
2 6 6 5 4 3 2 1 5 2 4 1 5 3	5 4 3 2 1 0 2 1 0 1 0

Example Explanation

For the second test case, **N**=5 and the initial sequence is 24153. In the first step, the number '1' is swapped 2 times and the sequence becomes 12453. In the second step, the number '5' is swapped 1 time and the sequence becomes 12435. In the third step, no swap is required as the number '2' is already in the 2nd position (0 swap). In the fourth step, the number '4' is swapped 1 time and the sequence becomes 12345. Lastly, no swapped is required in the 5th step (0 swap).

E The Escape

Time Limit | **2 seconds**

The legendary orb of Zot lies at the bottom of a 27 floor dungeon, guarded by deadly traps and horrible monsters. You, the brave hero, have fought your way to the bottom of this dungeon. You have killed every monster and destroyed every trap. Now you stand in the chamber with the orb. Many have died trying to reach this room, but you have succeeded. Not hesitating, you pick up the orb and put it in your bag. As you lift the orb from its resting place, it makes a horrible scream, and the dungeon begins to shake. The terrible lords of Pandemonium have noticed your crime. They gather their power to punish you. If you stay in the dungeon too long, they will reach inside your mind with their disgusting magic tentacles and squeeze your brain out through your ears. You must escape with the orb before this happens.

The dungeon has 27 floors. The chamber with the orb is on the bottom floor (floor 27th). Each floor is considered to be a 50 by 50 grid. The bottom left square is labelled (0,0). A square in the grid is either a wall square (in which case you cannot move through it), or a floor square. You can move freely through floor squares. In addition to this, a floor square may contain a ladder leading up to the next floor, or a trapdoor connected to a ladder on the lower floor. It costs 1 arbitrary time unit to move from one floor square to another, and it costs 1 arbitrary time unit to use ladders up. You can move in any of the four standard directions, and you can also move diagonally (provided there is no wall in the space you are moving into). Trapdoors are locked from the bottom, so you cannot use a trapdoor to go down to a lower floor. Once you have climbed a ladder the trapdoor it links to locks behind you and you may not return to the floor you came from.

On the 27th floor there are 3 ladders up, and no trapdoors down. On the top floor (floor 1) there are 3 trapdoors down, and one ladder up (the exit). On every floor in between there are 3 ladders up and 3 trapdoors down. Ladders (except the exit on floor 1) and trapdoors come in pairs, and are labelled 0, 1, or 2. So, for example, ladder number 1 leads up to the square containing trapdoor number 1 on the floor above. A ladder or trapdoor cannot be in the same square as another ladder or trapdoor.

Input

The first line of input is a number indicating the number of test cases (at most 5 test cases). Each test case is a block of 27 lines. The format is as follows:

The first line of a test case begins with a pair of numbers representing the starting coordinate of the hero floor 27. Following this are 3 pairs of numbers. These represent the coordinates of the ladders 0, 1, and 2 on floor 27 respectively. Following this is a single number. This tells you how many wall squares are on this floor. After this come pairs of coordinates representing the positions of the wall squares.

Lines 2-26 of a test case have this format: The line begins with 3 pairs of numbers representing the coordinates of the 3 trapdoors down. Following this are 3 pairs of numbers representing the coordinates of the ladders up. Following this is a number representing the number of wall squares, and after this are pairs of numbers representing the coordinates of the wall squares.

Line 27 of a test case has this format: The line begins with 3 pairs of number representing the coordinates of the 3 trapdoors down. Following this is a pair representing the location of the ladder exiting the dungeon. Following this is a number representing the number of wall squares, and pairs of numbers representing the coordinates of the wall squares as usual.

Output

For each test case output a single number representing the smallest number of moves it takes the hero to get to the exit ladder (you should *not* include the move it takes to use the exit ladder). It is guaranteed that there is an escape route (after all, you have walked that route down to the 27th floor)

Example (Simplified example for a 5 by 5 dungeon with 3 floors.)

Input	Output
1 0 0 4 4 3 1 3 4 3 1 2 2 2 3 2 0 0 4 0 0 4 2 2 4 4 0 1 5 2 1 3 3 2 0 1 1 1 2 0 0 0 2 3 4 4 4 5 0 1 1 0 2 0 3 0 4 0	10

Explanation

Floor 3 (start)

			L2	L0
	W	W	W	
			L1	
S				

Floor 2

T2				L1
			W	
	W	L0		
L2	W	W		
T0		W		T1

Floor 1 (exit)

			T2	E
	W			
T1	W			
W	W			
T0	W			

In this example the shortest path is obtained by using L0 then L2, for a total cost of 10 (8 squares and 2 ladders).

F Carrot Farm

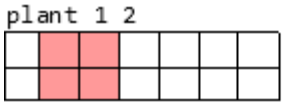
Time Limit | **3 seconds**

Farmer Yam just bought a piece of land that she will turn into a carrot farm. The land is a rectangle **W** ($1 \le W \le 10^8$) units wide and **L** ($1 \le L \le 10$) units long. Yam likes to think of it as an integer grid. The cells are numbered 0, 1, ..., **L** – 1 through the length and 0, 1, ..., **W** – 1 through the width. As a big planner, Yam has formulated a sequence of actions for her farming endeavor. Her sequence has length **n** ($1 \le n \le 50{,}000$). You will help Yam simulate her plan so she can answer some questions.

There are only *two* kinds of actions in Yam's plan: 1. plant carrot seeds or 2. harvest from existing plantations. The land is initially empty. Yam may mix the actions as she likes. The details of each action are as follows (you may also find sample input/output useful):

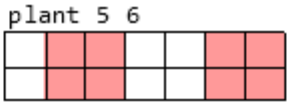
The *plant* action is to plant seeds between two given cell numbers on the wide side, occupying the full length of the land. If plantation areas touch, they will be merged into one. We promise that the new plantation area does **not** overlap with (though it may touch) any current plantation areas. After merging touching plantations (if any), you will report two numbers: the area of the unoccupied land immediately to the left of your most-recent plantation and separately the area of the unoccupied land immediately to the right of your most-recent plantation.

`plant 1 2`



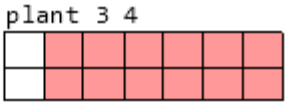
Answer: 2 8

`plant 5 6`



Answer: 4 0

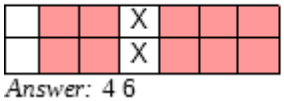
`plant 3 4`



Answer: 2 0

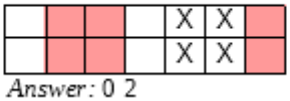
The *harvest* action is to harvest everything between two given cell numbers on the wide side, going the full length of the land. The harvest may split a plantation into two. We promise that every given harvest area will be completely contained within a plantation. You will report two numbers: the area of the planted land immediately to the left of the area just harvested and separately the area of the planted land immediately to the right of the area just harvested.

`harvest 3 3`



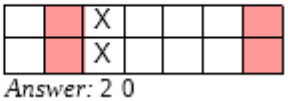
Answer: 4 6

`harvest 4 5`



Answer: 0 2

`harvest 2 2`



Answer: 2 0

Input

The first line contains the number of test cases **T** ($1 \le T \le 7$).

Each test case begins with a line containing *three* integers **W L n** as described above; the numbers are space-separated.

For that test case, each of the following **n** lines is of the format "action a b", where action is either *plant* or *harvest*, and **a** and **b** (**a** < **b**) specify the lower- and upper- cell numbers for that action, respectively.

Output

For each action, print on its own line the two numbers you are expected to report, separated by a single space. At the end of every test case, print two dashes "--" (without quote) followed by a new line character.

Example

Input	Output
2 7 2 7 plant 1 2 plant 5 6 plant 3 4 harvest 3 3 plant 3 3 harvest 4 5 harvest 2 2 4 5 1 plant 1 2	2 8 4 0 2 0 4 6 2 0 6 2 2 2 -- 5 5 --

G Dolphin

Time Limit	1 second

Dolphin is a well-known ice-breaking activity in Thailand. In order to play this game, everyone has to sit in circle. MC will select one person as a starting player. Each person, in clockwise order, has to shout the correct phrase in the dolphin sequence which is "1 dolphin, 1 jump, splash, 2 dolphins, 2 dolphins, 2 jumps, 2 jumps, splash, splash, 3 dolphins, 3 dolphins, 3 dolphins, 3 jumps, 3 jumps, 3 jumps, splash, splash, splash, ..." and so on. The game ends when there is one person who shout incorrect phrase. You know that you are **n**-th person to shout. As a programmer, you want to write a program to calculate the phrase you need to need to shout out.

Input

The first line will contain the number of test cases **T** ($1 \le T \le 1000$)

For each test case, there is an input **n** ($1 \le n \le 2\,000\,000\,000$)

Output

For each test case, print the **n**-th phrase in the dolphin sequence.

Example

Input	Output
5 1 2 3 4 100	1 dolphin 1 jump splash 2 dolphins 8 jumps

H

Glass Bridge

Time Limit	1 second

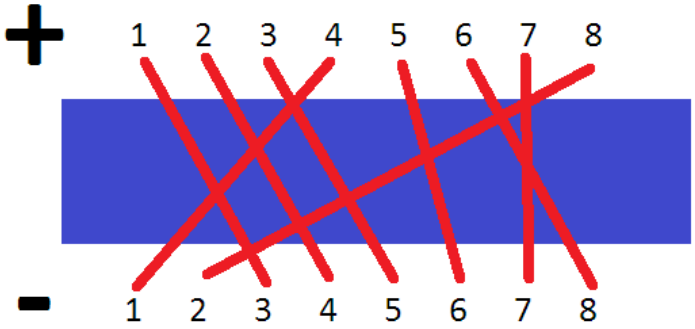
Water pollution is a problem common to many metropolis around the world, and the ACM metropolis of ICPC country is not an exception. Many cities try to solve this problem by having sewage treated before dumping into water sources, but for the ACM metropolis this is not a viable solution because space is a scarce treasure.

Alternatively, the government of ACM metropolis decided to use a chemical method. Specifically, the government would dump a treatment chemical into rivers around the metropolis to purify the water. However, the chemical reaction requires sunlight, thus all bridges across the rivers are being converted to Glass Bridge.

Funnily enough, the glass bridges still blocks 50% of the sunlight. If two glass bridges stacked, it only let 25% of total sunlight passing through. This could be a problem, as the chemical agent requires at least 40% of total sunlight to works. Although unavoidable, the government want to minimize the area that has less than 40% of sunlight.

The government has plans to construct **N** more bridge over a river. These bridges may overlaps, and thus limiting the sunlight. Given that three-story bridge is not feasible to construct (limiting the crossing bridges to two at a time), government wants to know how many areas of the river are covered with two bridge.

The metropolis around the river is structured such that the District 1 is directly across from District -1, District 2 from District -2, and so on. Each district has bridge to one another district across the river.



Input

First line has the number of test cases, **T** ($1 \le T \le 20$)

For each test case, the first line contains **N** specifying the number of bridge to construct. ($2 \le N \le 100\,000$)

Then on the next line are integers $a_1\ a_2\ a_3 \dots a_i \dots a_n$ ($1 \le a_i \le N$) saying that there are plan to construct bridge from District **i** to District $-a_i$.

Output

For each test case, output a number of area that has bridge crossing each other.

Example

Input	Output
2 8 4 8 1 2 3 5 7 6 3 3 2 1	10 3

I Rearrange

Time Limit	1 second

The code for `getMin( )` function is shown below. Given an integer `n` and an array of integers `arr`, you have to write the `rearrange( )` function which will be called by the `getMin( )` function so that your `rearrange( )` function minimize the return value from the `getMin( )` function.

```
int getMin(int n, int arr[]){
    arr2 = rearrange(arr);
    for(int i = 0; i < arr2.length(); i = i+1){
        n = n-arr2[i];
        if (n <= 0) break;
    }
    return n;
}
```

Your `rearrange( )` function should get an array of integers as an input and then return a new array of integers. The new array must contain the same members of the old array but the position of each member may be changed.

In this contest, the judges do not care about how you code the `rearrange( )` function. You just have to submit the minimum return value of the `getMin( )` function to win the contest.

Input

The first line will contain the number of test cases **T**. ($1 \le T \le 10$)

For each test case, there will be 2 lines. The first line of each test case contains 2 integers, **n** ($0 \le n \le 10\,000$) and **k** ($0 \le k \le 10\,000$), the length of array `arr`. The second line contains the members of array `arr`, `arr[0]`, ... , `arr[k-1]` ($0 \le arr[i] \le 1\,000\,000$). There will be exactly **k** integers in this line. Each integer is separated with a space.

Output

For each test case, print the minimum return value from `getMin( )` function, one test case per line.

Example

Input	Output
2 5 4 1 2 3 4 100 3 70 60 80	-3 -50

Explanation for sample test case 2: you send [70, 60, 80] to the `rearrange( )` function and it returns [80, 70, 60]. There can be more than one ways to rearrange the array `arr` in order to minimize the `getMin( )` function's return value.

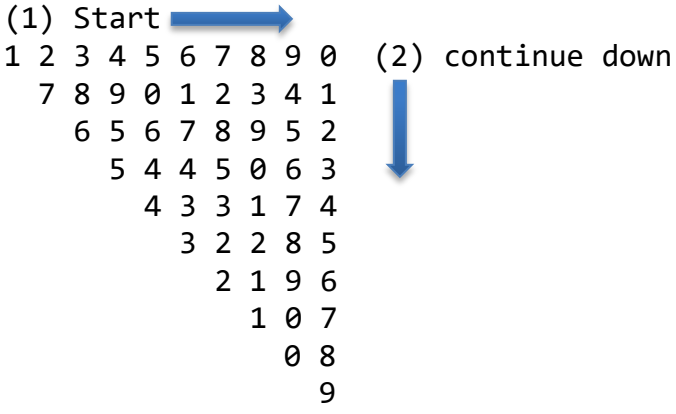
J

Cyclic Triangle

Time Limit | **5 seconds**

Given the area of the **N** × **N** right triangle grid, we want to fill them in with numbers. The **N** × **N** right triangle has the legs of length **N**. For the purpose of the explanation, we place one side of the leg horizontally, and the other side vertically as shown in the figure below. Note that, the left most column is considered the 0 column. The right most column is the **N**-1 column. And the top most row is considered the 0 row, and the bottom row is considered the **N**-1 row. For this problem, we want to fill the number in a circular manner. For the filling process, starting from the horizontal leg on the first row, at the opposite end of the right angle, then continue down vertically, and then move up diagonally to the left most of the second row. Then, the process continues starting from the second row, till there is no space in the triangle left to fill. The number that we must fill the will go from 0 to 9. However, we start the filling process from number 1 to 9, then restart from 0 and then continue to fill 1, 2, 3

For the case of **N** = 10, the resulting table is shown below. For this example, we filled in the triangle from (1) to (2) and then back toward (1) along the diagonal.



What is more interesting is the following. Once we know **N**, we can find out which number is filled in at a given row and column. From the example, where **N** = 10, if the position

is row = 0, and column = 7, then it is filled with number 8. However, if the position is row = 5 and column = 7, then the number is 2. Your task is to write a program that can quickly find out the circular number in the right triangle given the row and column.

Input

The first line will contain the number of test cases **T** ($1 \le T \le 10$)

For each test case, the input is (**Q** + 1)-line long. The first line of each test case contains two integer, **N** and **Q** (separated by a single space). **N** is the size of the triangle, and **Q** is the number of positions that you have to find out which number has been filled at each position. ($N \le 1\,000\,000$, $1 \le Q \le 10$).

For the next **Q** lines, each line contains two numbers, **R** and **C** (separated by a single space), where **R** is the interested row and **C** is the interested column. ($0 \le R \le C < N$) *Note that row and column is count from index 0.*

Output

For each interested position **R** and **C**, print out the filled in number in a single line, as shown in the output.

Example

Input	Output
2 12 5 7 8 0 0 11 11 5 10 2 7 1000000 6 7000 8000 5212 6824 700000 999999 5000 72913 900000 955555 8775 57412	2 1 3 6 1 8 0 0 4 8 3

Explanation

The figures in the table below show the triangles of **N** = 11, 12 and 13. You can use them to test the program.

11

```
1  2  3  4  5  6  7  8  9  0  1
   0  1  2  3  4  5  6  7  8  2
      9  1  2  3  4  5  6  9  3
         8  0  3  4  5  7  0  4
            7  9  2  6  8  1  5
               6  8  1  9  2  6
                  5  7  0  3  7
                     4  6  4  8
                        3  5  9
                           2  0
                              1
```

12 (The highlighted number is the results of the test cases in the sample input/output.)

```
1  2  3  4  5  6  7  8  9  0  1  2
   3  4  5  6  7  8  9  0  1  2  3
      2  7  8  9  0  1  2  3  3  4
         1  6  2  3  4  5  4  4  5
            0  5  1  8  6  5  5  6
               9  4  0  7  6  6  7
                  8  3  9  7  7  8
                     7  2  8  8  9
                        6  1  9  0
                           5  0  1
                              4  2
                                 3
```

13

```
1  2  3  4  5  6  7  8  9  0  1  2  3
   6  7  8  9  0  1  2  3  4  5  6  4
      5  3  4  5  6  7  8  9  0  7  5
         4  2  1  2  3  4  5  1  8  6
            3  1  0  0  1  6  2  9  7
               2  0  9  9  7  8  3  0  8
                  1  9  8  8  4  1  9
                     0  8  7  5  2  0
                        9  7  6  3  1
                           8  6  4  2
                              7  5  3
                                 6  4
                                    5
```

K

Organization

Time Limit	1 second

Elb Co., Ltd. has **N** employees divided into **k** nonempty teams. Obsessed with team dynamics and numerical metrics, it turns everything into numbers:

- Every employee **a** has undergone two tests, resulting in two numerical values x_a and y_a.
- The difference between employees **a** and **b** is given by

$$D(a, b) = | x_a - x_b | + | y_a - y_b |.$$

- The company's strength index (SI) is measured as

$$SI = \min \{ D(a, b) : a \text{ and } b \text{ are in different teams} \}$$

Next year, the company will continue to work with **k** teams but will arrange their employees to maximize the strength index. In this task, you will calculate the largest-possible strength index **SI** from **k** nonempty teams.

Input

The first line of input contains an integer, **T**, representing the number of test cases. (1 ≤ **T** ≤ 10)

Following that are **T** test cases. The first line of a test case contains two numbers: **N** and **k** (2 ≤ **k** ≤ 10, **k** ≤ **N** ≤ 1000).

Then, for the next **N** lines, each line represents an employee and contains this employee's test results **x** and **y**, separated by a single space (0 ≤ **x, y** ≤ 100 000).

Output

For each test case, print the largest-possible strength index SI, followed by a new line character.

Example

Input	Output
2 3 2 0 0 2 2 3 2 6 2 0 1 0 0 1 0 2 2 2 3 3 2	4 3

Example Explanation

From example 1, we are given 3 people, and **k** = 2. Note that, there are 3 ways of arranging, since each team must have at least 1 person. All cases are listed below.

Case 1: {(0,0)} {(2,2),(3,2)}, SI_1 = min{ D ((0,0), (2,2)), D ((0,0), (3,2))} = min{ 4, 5 } = 4

Case 2: {(0,0),(2,2)} {(3,2)}, SI_2 = min{ D ((0,0), (3,2)), D ((2,2), (3,2))} = min{ 5, 1 } = 1

Case 3: {(0,0),(3,2)} {(2,2)}, SI_3 = min{ D ((0,0), (2,2)), D ((3,2), (2,2))} = min{ 5, 1 } = 1

Hence, the **largest SI is SI_1 = 4**.

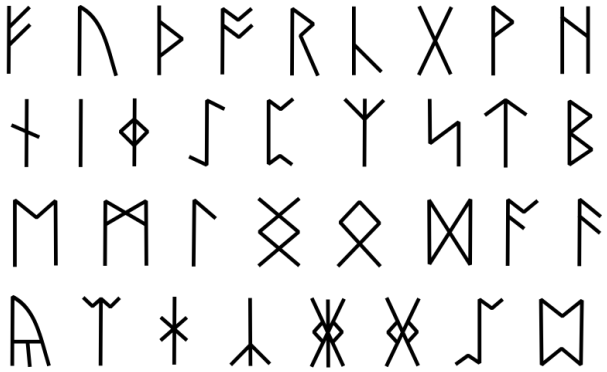
For example 2, we are given 6 people (0,1), (0,0), (1,0), (2,2), (2,3), (3,2), and **k** = 2. If you try to generate all cases, there will be 31 total possible assignments. If you calculate all the SI values, you will see that, the best team assignment is {(0,1), (0,0), (1,0)} and {(2,2), (2,3),(3,2)}. For this team assignment, its **SI is 3**. This is attained by D((0,1), (2,2)) pair or the D((1,0), (2,2)) pair.

L

Rune

Time Limit	1 second

Akara is a great wizard of Sanctuary. She is currently interested in runes, letters from an ancient language, which can enhance magic items and spells. Her recent studies show that some runes are more powerful than others.



Runes from Anglo-Saxon language.

Rune's power can be determined by a number of vowels (a, e, i, o, u) in its name. However, adjacent vowels will not increase its power more than once. For example:

1. Rune named ‘gattaca’ has a power of 3. Since there are 3 non-adjacent vowels in its name.
2. Rune named 'beautiful' also has a power of 3. Even though there are 5 vowels, because ‘eau’ are adjacency vowels and are counted as one.

Different languages have theirs own runes and cannot be mixed. Akara wants to rank her known runes from each language. So she can easily pick them up later. Your task, as her apprentice, is to write a program that takes lists of runes. Then print out every rune in that language from the most powerful to the least ones. If some of them fall in the same power level, show them in lexicographical order.

Input

The first line of input gives integer **L**, a number of languages you interested ($1 < L \leq 10$)

Next **L** lines will start with **N** ($1 < N \leq 10^5$), a number of runes in that language.

Following by **N** space-separated strings **W** indicated each rune. Each string **W** will contain only lowercase English alphabet (a-z) and has a maximum length of 100 characters.

Output

Output **L** lines. Each line contains **N** strings, separated by a single space. The strings must be ordered by its calculated rune's power. If some of them have the same power, order them lexicographically.

Example

Input	Output
2 2 gattaca beautiful 7 aem un ogla ulk ohm zaradar ng	beautiful gattaca zaradar ogla aem ohm ulk un ng