acm International Collegiate Programming Contest

第43屆ACM國際大學程式設計競賽－亞洲區台北站

The 43rd Annual ACM International Collegiate Programming Contest - Asia Taipei Regional Contest

Taiwan Online Programming Contest

- Problems: There are 9 problems in this contest. (24 pages including cover).
- Input: Input of all problems are from the standard input. Multiple inputs will be given according to the instructions of each problem.
- Output: All output should be directed to the standard output.
- Time limit: Judge will run each submitted program with a certain time limit (given in the table below).
- Please note that if your program exceeds the memory limit, you may receive **any verdict other than** `Yes` (including `Wrong-Answer`, `Runtime-Error`).

Problem	Problem Name	Time Limit	Memory Limit
A	Accepted or not?	1 s	256 MB
B	RPG	1 s	256 MB
C	Minimum perimeter	1 s	256 MB
D	Arrow Game	1 s	256 MB
E	XOR or AND?	2 s	256 MB
F	Discount	1 s	256 MB
G	Hexcraft	2 s	256 MB
H	Scoreboard	1 s	256 MB
I	Fency Cows	1 s	256 MB

This page is intentionally left blank.

A. Accepted or not?

You may not know that in this contest, most of the problems are settled done just before the contest starts!

Since the problems are not prepared by the same person, the formats of input/output may differ from problem to problem. Even worse, the response formats of each problem are also diverse. For example, an author may code a judge program responding `"AC"` for accepted solutions and `"NA"` for not accepted solutions. While another author may response `"Accepted"` for accepted solutions and `"Not even close!"` for not accepted solutions. Thus, it's hard for the chief judge to write a script to give the correct verdict to contestants.

Therefore, the chief judge sets rules to decide the verdict as follow:

1. Replace all the uppercase characters into lowercase.
2. If the response string contains `'n'`, it's not accepted.
3. Otherwise, if the response string contains `'ac'` or `'yes'` as consecutive characters, it's accepted.
4. Otherwise, it's unknown. And, the author will be blamed by the chief judge.

The contest has just started as the chief judge came up with those rules. He doesn't have enough time to code it as a script (Yes, 5 hours is not even enough). He promises that if you can code it out for him (and for the contest). He will be very happy and give you an accepted!

Input

The first line of input contains an integer T indicating the number of responses from judge programs.

Each of following T lines contains a string S indicating the response string.

- $1 \leq T \leq 10^3$
- $1 \leq |S| \leq 20$,
- S consists of lowercase and uppercase English letters (`'a-zA-Z'`).

Output

For each response string, if the verdict is accepted, please output "Yes" in one line (without quoting marks). If the verdict is not accepted, please output "No" in one line. Otherwise, please output "Unknown" in one line.

Sample Input 1	Sample Output 1
10 AC NA Accepted NotEvenClose yEs eyes nuh nie tak accenped	Yes No Yes No Yes Yes No No Unknown No

B. RPG

You, the hero, are going to fight with the final boss of the world. As a hero, you have three character "abilities": constitution, dexterity, and intelligence. Your current characteristics are C constitution, D dexterity, and I intelligence.

To make you even stronger, the villagers give you N universal ability potions, where each potion can be used to increase any ability by one. According to an elder scroll, the larger is your overall strength, the higher is the probability you can beat the boss. The overall strength is calculated as the geometric mean of the abilities. Please find the optimal way to utilize the potions, so the overall strength is maximized!

Input

The first line contains an integer T indicating the number of testcases. Following T lines each contains four space-separated integers N, C, D, I.

- $1 \leq T \leq 200000$
- $0 \leq N, C, D, I \leq 10^6$

Output

For each testcase, please output one line containing only one integer indicating the cube of the maximum overall strength.

Sample Input 1	Sample Output 1
3 3 1 1 1 2 4 6 8 1 10 22 23	8 288 5566

This page is intentionally left blank.

C. Minimum perimeter

On a 2D plane, there are n distinct rays shot from the origin $(0, 0)$. A point $p = (x, y)$ is picked.

Then you have to pick one point on each ray and draw the Convex Hull of p and these n points. Your job is to pick these n points so that the resulting polygon has the minimum perimeter possible.

A polygon is convex if every segment with endpoints inside the polygon lie entirely in the polygon (may touch the boundary).

A Convex Hull of a set of points $\mathcal{S}$ is defined as the convex polygon with minimum area that has every point of $\mathcal{S}$ inside or on the boundary.

The perimeter of a polygon degraded into a segment is twice the segment's length.

Input

The first line contains an integer T indicating the number of testcases. Each testcase consists of $n + 1$ lines of input. The first line of a testcase consists of three integers n, x, y. The $(i + 1)$-th line of a testcase contains two integers x_i, y_i describing the ith ray.

- $1 \leq T \leq 100$
- $2 \leq n \leq 100$
- $0 \leq x, y, x_i, y_i \leq 10^9$
- $0 < x^2 + y^2, x_i^2 + y_i^2$
- n rays are given in counter-clockwise order.
- $p = (x, y)$ lies strictly inside the sector bounded by (x_1, y_1) and (x_n, y_n).

Output

For each testcase, please output one line containing only one real number for the answer. The output will be considered correct if the relative error to the answer is less than 10^{-6}.

Sample Input 1	Sample Output 1
2 2 1 1 1 0 0 1 2 2 2 2 1 1 2	2.828427125 3.394112550

D. Arrow Game

Besides delicious street foods, there are many games to play in a night market! One day, Drazil discovers a new game in a night market named arrow game. This is how the game is played:

Before the game starts, a grid of size $N \times M$ is presented to the player. There are two types of cells in the grid. The first type is the arrow cell. An arrow cell contains an arrow pointed up, down, right or left. The second type is the score cell. Each score cell carries a positive integer score. There exists at least one score cell in the grid.

To start the game, the player presses a button and a starting cell is picked uniformly from all $N \times M$ cells. Then, from the picked cell, the player moves a token following the arrow directions. If the token ended up outside the grid boundary or looping forever, the player gets 0 scores. Otherwise, the token stopped at a score cell. In this case, the score of that cell becomes the player's score.

This game may seem based on luck only. To increase playability, the game master introduced a new step to the gameplay. Now, a player can change the direction of at most one arrow cell before the starting cell is picked!

Drazil desires to know what the highest expected score could be after at most one change. Can you help him to calculate this value?

Input

Each cell in the grid is represented by a character for compactness of the input. For arrow cells, `u`, `d`, `r` and `l` represent up, down, right and left arrows respectively. For score cells, first capital English letters starting from `A` are used as labels. No two score cells share the same label.

The first line of the input contains an integer T indicating the number of testcases. Following are T testcases. The first line of each testcase consists of three positive integers N, M, and S. Those integers denote the size of the grid and the number of score cells in the grid respectively. The second line of each testcase contains S positive integers C_1, C_2, ..., C_S. C_i is the score of the cell labeled by the i^{th} capital English letter. Next N lines of

each testcase contain a string of length M denoting the grid. Those strings only consist of `u`, `d`, `r`, `l`, and the first S capital English letters.

- $1 \leq T \leq 150$
- $1 \leq N, M, N \times M \leq 5 \times 10^5$
- $\sum N \times M \leq 3 \times 10^6$
- $1 \leq C_i \leq 10^8$
- $1 \leq S \leq \min(N \times M, 26)$

Output

For each testcase, please output one line containing only one integer indicating the maximized expected score multiplied by $N \times M$. Please note that this value must be an integer in this problem.

Sample Input 1	Sample Output 1
2 1 3 2 100 200 AuB 3 1 1 100 d u A	500 300

E. XOR or AND?

Eddy likes to play with integers. He has collected N integers to play with.

Someday, Eddy learns something about bit-operations. Those operations are defined on non-negative integers. Specifically, Eddy has learned three bit-operations XOR, OR, AND. They work as follow:

- a XOR $b = c$, where i-th bit (in binary representation) of c will be 0 if i-th bit of a and b are the same; otherwise, it will be 1.
- a OR $b = c$, where i-th bit of c will be 0 if i-th bit of a and b are both 0; otherwise, it will be 1.
- a AND $b = c$, where i-th bit of c will be 1 if i-th bit of a and b are both 1; otherwise, it will be 0.

Eddy thinks these operations are very funny and want to take out his collected integers to play with them. To do so, Eddy needs to distribute those N integers into three disjoint sets A, B, C. For each set, he will permute its elements as a sequence S_A, S_B, S_C, respectively. Then, the satisfaction of this assignment will be $\mathsf{XOR}(S_A)+\mathsf{OR}(S_B)+\mathsf{AND}(S_C)$, where $\mathsf{OP}(S)$ is defined as $(((S_0$ OP $S_1)$ OP $S_2)\ldots)$. If the sequence is empty, no matter which operation is, the value will be defined as 0.

Eddy wants to find a way to distribute his integers such that the satisfaction is maximized. However, he has no idea about how to find it out. Please find an assignment for Eddy to cheer him up!.

Input

The first line of input contains an integer N indicating that Eddy has collected N integers. The second line contains N space-separated integers e_i indicating the integer Eddy collected.

- $1 \leq N \leq 511$
- $0 \leq e_i \leq 511$

Output

Please output four lines.

The first line contains a non-negative integer indicating the maximum possible satisfaction.

The first line contains the number of elements in S_A followed by its element separated by a single space.

The second line contains the number of elements in S_B followed by its element separated by a single space.

The third line contains the number of elements in S_C followed by its element separated by a single space.

Sample Input 1	Sample Output 1
3 1 2 3	6 2 1 2 0 1 3

Sample Input 2	Sample Output 2
4 511 256 256 256	1023 1 256 1 511 2 256 256

F. Discount

Dreamoon likes to travel a lot. He is planning to go to Incredible Convenient Purchasing Country (ICPC) next. ICPC is famous for the high density of convenience stores. They are so popular that the nearest one must be within 100 meters away no matter where you are in ICPC. Dreamoon wants to make good use of his time and his money in ICPC buying many things!

Dreamoon discovered there is a custom when buying things in ICPC after some research. The original price of any goods usually being as high as 10^{10^6}. Of course, Dreamoon is not going to pay such a high price. Instead, you can remove any two consecutive distinct digits from the original price! Also, you can perform the remove operation as many times as you want as long as every removal is valid.

For example, if the original price is 123, one could pay 1 dollar by removing 23 or pay 3 dollars by removing 12. However, paying 2 dollars is illegal since 1 and 3 are not consecutive digits. Moreover, if the original price is 111, no removal is valid since all consecutive digits are the same. Also, if the original price is 5566, one could get it for free by removing 56 twice!

Please be aware of one special case. If leading zeros occur during this process, they will not be removed automatically. For example, removing 10 from 100003 results in 0003, not 3. Of course, you can remove 03 after that and pay 0 dollars.

Dreamoon wants to pay as little money as possible. To achieve perfection in removing digits, he is doing lots of practices right now. Your task is to find out the lowest possible price after removals so Dreamoon can check if he is correct.

Input

The first line of the input contains an integer T indicating the number of testcases. Following T lines each contains a testcase. Each testcase consists of one positive integer P denoting the original price. P does not have any leading zeros.

- $1 \leq T \leq 100$
- $0 \leq P \leq 10^{10^6}$
- sum of the length of all prices $\leq 10^7$

Output

For each testcase, please output one line containing only one integer indicating the lowest possible price after removals.

Sample Input 1	Sample Output 1
5 123 5566 103 41315 100000000000000000023	1 0 1 3 0

G. Hexcraft

Welcome to the Best Game Corporation! We are glad you are on board with us. Due to the lack of programmers, the development of our new title, “hexcraft”, is seriously delayed. Your first job is to write an important piece of the game, the pattern editor.

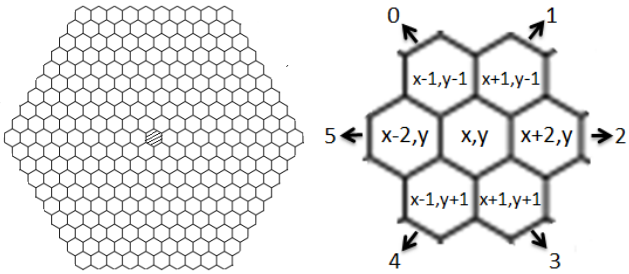
Pattern Editor

As its name might suggest, the game is mainly about creating patterns in a hexagonal grid. Simply put, the goal of a user is to select some cells to be black and leave the unselected cells to be white. We have designed an easy-to-use editor which consists of four commands to aid users exploring unlimited possibilities.

We will first describe the coordinate system of the hexagonal grid and then document the details of those four operations that you are about to implement.

Edit Area and Coordinate System

To be specific, the edit area is a hexagonal grid of which each side has 10 cells. The coordinate of the center cell is $(0, 0)$. The left image below is a sample of such a grid with the center cell filled with stripe pattern.



The coordinates of the rest of the cells are defined in a relative manner. The right figure above demonstrates the coordinates of neighboring cells relative to a cell located at (x, y). Also, six directions, indexed from 0 to 5, are defined as the right figure above.

Lastly, all the cells are set to unselected initially.

Operations

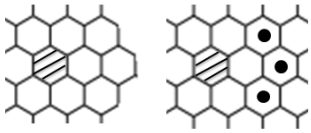
We will first give the definition of two terms.

The component of a selected cell, or a selected cell's component, refers to all selected cells which connect (directly or indirectly) to that selected cell. Two selected cells are connected if and only if they share an edge. Please note that unselected cells are not connected to any selected cells.

A coordinate is invalid means either it is outside of the edit area or it does not exist at all. For example, $(1, 0)$ and $(100000, 0)$ are both invalid coordinates.

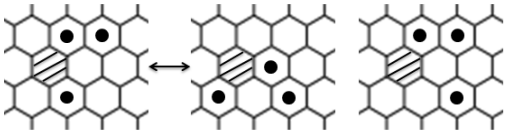
Now, we give detailed descriptions of the four operations that should be implemented. Each operation will cause some changes to the grid. Also, each operation will produce an integer as a kind of logging message.

1. $Create\ x\ y\ w$
   Parameters x, y, and w indicate a sub hexagonal grid in the edit area. Such sub grid has w cells on each side and is centered at (x, y).
   If any of the cells (including (x, y) itself) in the sub grid is not valid or is already selected, this operation is invalid. Otherwise, all cells in the sub grid are selected.
   If this operation is invalid, log -1. Otherwise, log the number of cells selected.
2. $Remove\ x\ y$
   If (x, y) is not valid or is not selected, this operation is invalid. Otherwise, all cells in the component of (x, y) are unselected.
   If this operation is invalid, log -1. Otherwise, log the number of cells unselected.
3. $Move\ x\ y\ dir\ len$
   If (x, y) is not valid or is not selected, this operation is invalid. Otherwise, all cells in the component of (x, y) are moved along the direction dir for len units long unless blocked described as follows. If during the moving process, any cell in the moving component hits (connects to) any other selected cells or reaches the boundary of the edit area, the moving is stopped.
   Images below further explain the blocking mechanism. The striped cell in the left image can only be moved 2 units long along direction 2 due to the boundary. The striped cell in the right image can only be moved 1 unit long along direction 2 if one of the three dotted cells is selected.



If this operation is invalid, log -1. Otherwise, log the distance actually moved.

4. *Rotate* x y c
   If (x, y) is not valid or is not selected, then this operation is invalid. Otherwise, if c is 0, rotate the component of (x, y) clockwise centered at (x, y) by 60 degrees. If c is 1, rotate counter-clockwise. When rotating a component, we ignore all other selected cells that would possibly become obstacles otherwise. However, after rotation, if some of the rotated cells' locations are invalid or some of the rotated cells overlap some already selected cells (that are not part of the rotating component), this operation is also invalid (and has no effect).
   Images below further explain the rotate operation. The left and middle images are the rotation results centered at the striped cell with all dotted cells selected. In the right image, the clockwise rotation centered at the striped cell is invalid due to overlapping.



If this operation is invalid, log -1. Otherwise, log the number of cells rotated.

Now, to verify the correctness of your implementation, quality assurance team prepared a sequence of operations for you to simulate. You should apply these operations sequentially and print the log of these operations. In the end, the status of the edit area should also be printed.

Input

The first line of the input contains an integer T indicating the number of operations. Following T lines each contains an operation following the documentation given above.

- $1 \leq T \leq 1000$

- $-20 \leq x \leq 20$
- $-10 \leq y \leq 10$
- $0 \leq dir \leq 5$
- $1 \leq len, w \leq 10$
- $0 \leq c \leq 1$

Output

For each operation, please output one line containing only one integer indicating the log of this operation.

After the logs, please output 19 lines each contains 37 characters. The j-th character of the i-th line is

- ' ' if $(j-19, i-10)$ is invalid.
- '.' if $(j-19, i-10)$ is unselected.
- 'x' if $(j-19, i-10)$ is selected.

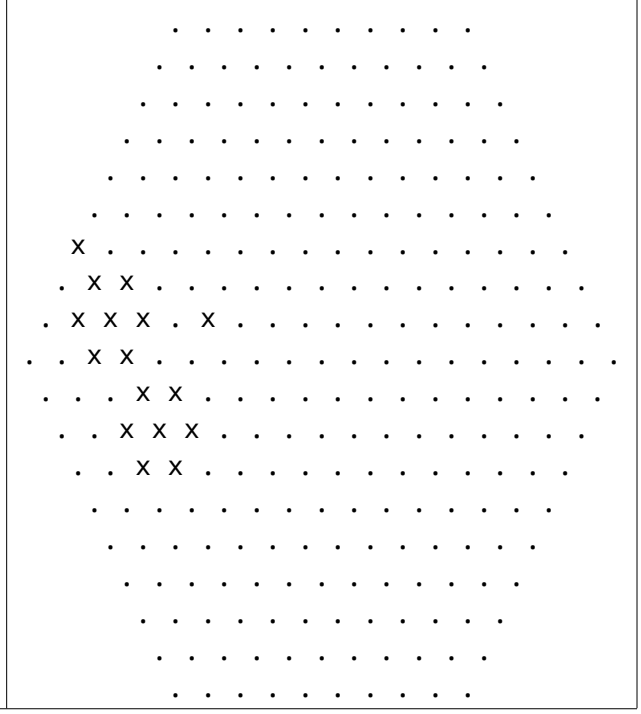
Please note that trailing spaces are necessary, each line of the final 19 lines should contain exactly 37 characters.

Sample Input 1

```
11
Create 0 1 3
Create 0 0 2
Create 3 -3 1
Move 1 1 1 4
Rotate 2 -2 1
Move 2 -2 5 10
Create 8 8 2
Remove 9 7
Create -10 2 2
Create -7 -1 1
Rotate -13 -1 1
```

Sample Output 1

```
-1
7
1
1
8
8
7
7
7
1
-1
```



This page is intentionally left blank.

H. Scoreboard

Paul enjoys following ICPC contests. However, just watching the scoreboard page could be very boring, so Paul comes up with a problem: at least how many additional accepted solutions are required so that, on the resulting scoreboard, each team dominates all other teams who are ranked behind them.

Team A dominates team B if the set of problems solved by team B is a subset of the set of problems solved by team A. That is, all the problems solved by team B are also solved by team A.

Now, a contest is in the progress and the scoreboard is given. Can you help Paul find the answer?

Input

The first line contains an integer T indicating the number of testcases.

For each testcase, the first line contains two space-separated integers P, N indicating the number of problems and the number of teams. The scoreboard is given in the following N lines. The i-th line contains a `"01"`-string of length P indicating the status of the team currently in i-th place; the j-th character is 1 if the team already solved the j-th problem, and 0 otherwise.

The teams are given in the order as shown on the scoreboard (in the progress). As all the ICPC contests, there are $8 \sim 14$ problems in each contest.

- $1 \leq T \leq 100$
- $8 \leq P \leq 14$
- $3 \leq N \leq 1000$

Output

For each testcase, please output one line containing only one integer indicating the minimum number of additional accepted solutions required.

Sample Input 1	Sample Output 1
2 8 7 101111100 101011100 001111000 001111000 001100000 000100000 000000000 8 7 101111101 101010100 001101000 000111000 001100000 000100000 000000000	1 3

I. Fency Cows

Cows are very fancy nowadays. There are C cows on the field, where the field can be represented as a two dimensional plane. These cows are numbered from $1, 2, \ldots, C$, and each cow has a location (x_i, y_i) and a fancy value v_i. The fancy value is determined by their coordinator – Farmer John. These values may be non-negative or negative.

Farmer John wants to build a fence that protects his valuable cows. However, the fence he built must be fancy enough for his fancy cows. He designated N places for the fence's corner posts. He would like to use some of the corner posts and build a fence that looks like a *non-empty convex polygon* on the plane, with all its corners on the corner posts. A convex polygon is a region that for any given two points inside the polygon, the straight line segment connecting these two points never intersects with the boundary of the polygon. A non-empty convex polygon means that the area of this polygon must be positive.

Given the coordinates of the cows and the corner posts, can you find the best way to build the fence such that the sum of all cow's fancy values inside the fence achieves the maximum?

Input

In the first line of each test, there are two integers C and N. Then C lines follow, each line contains three integers x_i, y_i, v_i, representing the location and the fancy value for each cow. Then N lines follow. Each line has two integers p_i and q_i denoting the coordinates of the i-th corner post.

- $1 \leq C \leq 10000$.
- $3 \leq N \leq 300$.
- $-10000 \leq v_i \leq 10000$.
- $0 \leq x_i, y_i, p_i, q_i \leq 100\,000\,000$.
- It is guaranteed that no cow will appear on any straight line that passes through any two corner posts. No two objects (either corner posts or cows) share the same position. No three corner posts will be collinear.

Output

For each test case, please output an integer indicating the maximum possible achievable sum of the fancy values.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

```
2 3
1 2 -1
2 1 3
0 0
0 5
5 0
```

Sample Output 2

```
2
```

Sample Input 3

```
1 3
1 2 -1
0 0
0 5
5 0
```

Sample Output 3

```
-1
```