Problem A. Namomo Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

“gshfd1jkhaRaadfglkjerVcvuy0gf” said Prof. Pang.

To understand Prof. Pang’s word, we would like to calculate the number of *namomo subsequences* of it. The word by Prof. Pang is a string s with n characters where each character is either an English letter (lower or upper case) or a digit. The i-th character of s is denoted by $s[i]$ $(1 \leq i \leq n)$. A subsequence t of s is defined by a list of indices $t_1, \ldots, t_6$ such that $1 \leq t_1 < t_2 < \ldots < t_6 \leq n$. Let $compare(c_1, c_2)$ be a function on two characters such that $compare(c_1, c_2) = 1$ when $c_1 = c_2$ and $compare(c_1, c_2) = 0$ otherwise. t is a namomo subsequence of s if and only if for any $1 \leq i < j \leq 6$, $compare(s[t_i], s[t_j]) = compare(namomo[i], namomo[j])$, where $namomo[x]$ represents the x-th character of the string “namomo” $(1 \leq x \leq 6)$.

Output the number of namomo subsequences of a given string s modulo 998244353.

Input

The first line contains a string s with n characters $(6 \leq n \leq 1000000)$. s contains only lower case English letters (‘a’ – ‘z’), upper case English letters (‘A’ – ‘Z’) and digits (‘0’ – ‘9’).

Output

Output one integer – the answer modulo 998244353.

Examples

standard input	standard output
`wohaha`	`1`
`momomo`	`0`
`gshfd1jkhaRaadfglkjerVcvuy0gf`	`73`
`retiredMiFaFa0v0`	`33`

Problem B. Rectangle Flip 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Prof. Pang enters a trap room in a dungeon. The room can be represented by an n by m chessboard. We use (i, j) $(1 \le i \le n,\ 1 \le j \le m)$ to denote the cell at the i-th row and j-th column. Every second, the floor of one cell breaks apart (so that Prof. Pang can no longer stand on that cell.) After nm seconds, there will be no cell to stand on and Prof. Pang will fall through to the next (deeper and more dangerous) level.

But Prof. Pang knows that calm is the key to overcome any challenge. So instead of being panic, he calculates the number of rectangles such that every cell in it is intact (i.e., not broken) after every second. (A rectangle represented by four integers a, b, c and d $(1 \le a \le b \le n, 1 \le c \le d \le m)$ includes all cells (i, j) such that $a \le i \le b, c \le j \le d$. There are $\frac{n(n+1)}{2} \times \frac{m(m+1)}{2}$ rectangles in total.)

Input

The first line contains two integers n, m $(1 \le n, m \le 500)$ separated by a single space.

The $(i + 1)$-th line contains two integers a, b separated by a single space representing that the cell (a, b) breaks apart at the i-th second. It is guaranteed that each cell appears exactly once in the input.

Output

Output nm lines. The i-th line should contain the number of rectangles such that every cell in it is intact after the first i cells break apart.

Example

standard input	standard output

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

Note

In the example, after the first second, there are 3 rectangles of area 1 and 2 rectangles of area 2 that satisfy the constraint. So the first line should contain a 5. After the second second, only cells in the second column remains intact. The answer should be 3. After the third second, only one cell remains intact. The answer should be 1. After the fourth second, all cells broke apart so the answer should be 0.

Problem C. Random Shuffle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Prof. Pang is selecting teams that advance to the world final contest. As the regionals are cancelled, he uses random shuffle to rank the teams. There are n teams in total. His code is as follows:

```
uint64_t x;//uint64_t represents 64-bit unsigned integer
int n;
uint64_t rand() {//this is a xor-shift random generator
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    return x;
}
int main() {
    cin >> n;
    cin >> x;
    for (int i = 1; i <= n; i++) {//random shuffle [1, 2,..., n]
        a[i] = i;
        swap(a[i], a[rand() % i + 1]);
    }
    for (int i = 1; i <= n; i++) {//print the result
        cout << a[i] << (i == n ? ’\n’ : ’ ’);
    }
}
```

He compiled and ran his code and then entered n and some special nonnegative integer x. He printed the result on paper.

One day later, Prof. Pang forgot his choice for x. You are given the result of the code and the integer n. Please recover the number x that Prof. Pang had entered.

Input

The first line contains a single integer n ($50 \le n \le 100000$) – the number of teams.

The next line contains n integers – the result printed by Prof. Pang's code. It is guaranteed that the result is correct, i.e., there exists an integer x ($0 \le x \le 2^{64} - 1$) that leads to the result.

Output

Output the integer x ($0 \le x \le 2^{64} - 1$) Prof. Pang had entered. If there are multiple possible x's, print any one.

Example

standard input

50
36 22 24 21 27 50 28 14 25 34 18 43 47
13 30 7 10 48 20 16 29 9 8 15 3 31 12
38 19 49 37 1 46 32 4 44 11 35 6 33 26
5 45 17 39 40 2 23 42 41

standard output
16659580358178468547

Note

Note that the second line of the sample input is wrapped to fit in the width of page.

Problem D. City Brain

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

Prof. Pang works for the City Brain program of Capital Grancel. The road network of Grancel can be represented by an undirected graph. Initially, the speed limit on each road is 1m/s. Prof. Pang can increase the speed limit on a road by 1m/s with the cost of 1 dollar. Prof. Pang has k dollars. He can spend any nonnegative integral amount of money on each road. If the speed limit on some road is am/s, it takes $1/a$ seconds for anyone to go through the road in either direction.

After Prof. Pang spent his money, Prof. Du starts to travel from city s_1 to city t_1 and Prof. Wo starts to travel from city s_2 to city t_2. Help Prof. Pang to spend his money wisely to minimize the sum of minimum time of Prof. Du's travel and Prof. Wo's travel. It is guaranteed that s_1 and t_1 are connected by at least one path and that s_2 and t_2 are connected by at least one path.

Input

The first line contains three integers n, m, k ($1 \le n \le 5000$, $0 \le m \le 5000$, $0 \le k \le 10^9$) separated by single spaces denoting the number of vertices, the number of edges in the graph and the number of dollars Prof. Pang has.

Each of the following m lines contains two integers a, b ($1 \le a, b \le n, a \ne b$) separated by a single space denoting the two endpoints of one road. There can be multiple roads between the same pair of cities.

The following line contains four integers s_1, t_1, s_2, t_2 ($1 \le s_1, t_1, s_2, t_2 \le n$) separated by single spaces denoting the starting vertices and ending vertices of Prof. Du and Prof. Wo's travels.

Output

Output one decimal in the only line – the minimum sum of Prof. Du's travel time and Prof. Wo's travel time. The answer will be considered correct if its absolute or relative error does not exceed 10^{-9}.

Examples

standard input	standard output

`6 5 1`
`1 2`
`3 2`
`2 4`
`4 5`
`4 6`
`1 5 3 6`
	`5.000000000000`

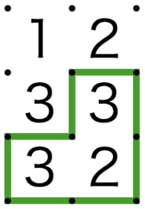
`1 0 100`
`1 1 1 1`
	`0.000000000000`

`4 2 3`
`1 2`
`3 4`
`1 2 3 4`
	`0.833333333333`

Problem E. Tube Master III

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Prof. Pang is playing "Tube Master". The game field is divided into $n \times m$ cells by $(n+1) \times m$ horizontal tubes and $n \times (m+1)$ vertical tubes. The product nm is an **even** number. There are $(n+1)(m+1)$ crossings of the tubes. The 2D coordinate of the crossings are (i, j) $(1 \leq i \leq n+1,\ 1 \leq j \leq m+1)$. We name the crossing with coordinate (i, j) as "crossing (i, j)". We name the cell whose corners are crossings $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$ as "cell (i, j)" for all $1 \leq i \leq n$, $1 \leq j \leq m$. Additionally, each cell (i, j) contains an integer $count_{i,j}$.



The above figure shows a game field with $n = 3, m = 2$ (the third sample).

Prof. Pang decides to use some of the tubes. However, the game poses several weird restrictions.

1. Either 0 or 2 tubes connected to each crossing are used.
2. There are exactly $count_{i,j}$ turning points adjacent to cell (i, j). A turning point is a crossing such that exactly 1 horizontal tube and exactly 1 vertical tube connected to it are used. A turning point (x, y) is adjacent to cell (i, j) if crossing (x, y) is a corner of cell (i, j).

It costs $a_{i,j}$ to use the tube connecting crossings (i, j) and $(i, j+1)$. It costs $b_{i,j}$ to use the tube connecting crossings (i, j) and $(i+1, j)$. Please help Prof. Pang to find out which tubes he should use such that the restrictions are satisfied and the total cost is minimized.

Input

The first line contains a single positive integer T denoting the number of test cases.

For each test case, the first line contains two integers n, m $(1 \leq n, m \leq 100)$ separated by a single space.

The i-th of the following n lines contains m integers $count_{i,1}, count_{i,2}, \ldots, count_{i,m}$ $(0 \leq count_{i,j} \leq 4)$ separated by single spaces.

The i-th of the following $n+1$ lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ $(1 \leq a_{i,j} \leq 10^9)$ separated by single spaces.

The i-th of the following n lines contains $m+1$ integers $b_{i,1}, b_{i,2}, \ldots, b_{i,m+1}$ $(1 \leq b_{i,j} \leq 10^9)$ separated by single spaces.

It is guaranteed that nm is an **even** number and that the total sum of nm over all test cases does not exceed 10^4.

Output

For each test case, output an integer that denotes the minimum cost. If there is no valid configuration, output "`-1`" instead.

Example

standard input

```
4
2 3
4 3 2
2 3 4
2 1 1
2 1 2
1 2 1
1 2 1 2
1 1 1 2
2 2
2 1
2 1
1 2
2 2
1 2
1 2 1
2 1 1
3 2
1 2
3 3
3 2
1 1
1 1
2 2
1 1
1 1 1
1 1 1
2 2 2
2 2
1 2
3 4
5 6
7 8
9 10
11 12 13
14 15 16
```

standard output

```
13
8
11
-1
```

Problem F. Rooks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Prof. Pang plays chess against his rival Prof. Shou. They are the only two players in the game. The chessboard is very large and can be viewed as a 2D plane. Prof. Pang placed n_1 rooks and Prof. Shou placed n_2 rooks. Each rook is a point with integer coordinates on the chessboard. One rook is *attacked* by another rook if they satisfy all of the following conditions:

- They are placed by different players.
- They have the same x-coordinate or y-coordinate.
- There is no other rook on the line segment between them.

Help Prof. Pang and Prof. Shou to decide which rooks are attacked.

Input

The first line contains two integers n_1, n_2 ($1 \le n_1, n_2 \le 200000$) separated by a single space denoting the number of rooks placed by Prof. Pang and the number of rooks placed by Prof. Shou.

The i-th ($1 \le i \le n_1$) line of the next n_1 lines contains two integers x, y ($-10^9 \le x, y \le 10^9$) separated by a single space denoting the location (x, y) of the i-th rook placed by Prof. Pang.

The i-th ($1 \le i \le n_2$) line of the next n_2 lines contains two integers x, y ($-10^9 \le x, y \le 10^9$) separated by a single space denoting the location (x, y) of the i-th rook placed by Prof. Shou.

It is guaranteed that the $n_1 + n_2$ rooks placed by the players are distinct (i.e., no two rooks can have the same location).

Output

Output a string with length n_1 on the first line. The i-th ($1 \le i \le n_1$) character should be 1 if the i-th rook placed by Prof. Pang is attacked and 0 otherwise.

Output a string with length n_2 on the second line. The i-th ($1 \le i \le n_2$) character should be 1 if the i-th rook placed by Prof. Shou is attacked and 0 otherwise.

Example

standard input	standard output

3 2
0 0
0 1
1 0
0 -1
-1 0
	100
	11

Problem G. Prof. Pang's sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Prof. Pang is given a fixed sequence $a_1, \ldots, a_n$ and m queries.

Each query is specified by two integers l and r satisfying $1 \le l \le r \le n$. For each query, you should answer the number of pairs of integers (i, j) such that $l \le i \le j \le r$ and the number of distinct integers in $a_i, \ldots, a_j$ is odd.

Input

The first line contains a single integer n $(1 \le n \le 5 \times 10^5)$.

The next line contains n integers $a_1, \ldots, a_n$ $(1 \le a_i \le n$ for all $1 \le i \le n)$ separated by single spaces.

The next line contains a single integer m $(1 \le m \le 5 \times 10^5)$.

Each of the next m lines contains two integers l and r $(1 \le l \le r \le n)$ separated by a single space denoting a query.

Output

For each query, output one line containing the answer to that query.

Examples

standard input	standard output

5
1 2 3 2 1
5
1 5
2 4
1 3
2 5
4 4
	10
	3
	4
	6
	1

5
2 3 5 1 5
5
2 3
1 1
1 3
2 5
2 4
	2
	1
	4
	6
	4

10
2 8 5 1 10 5 9 9 3 5
10
6 8
1 2
3 5
5 7
1 7
3 9
4 9
1 4
3 7
2 5
	4
	2
	4
	4
	16
	16
	12
	6
	9
	6

Problem H. Prof. Pang Earning Aus

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Prof. Pang has only 1 Au in his pocket. (Yes, Prof. Pang is from Austan and he uses the currency Au there.)

He will make use of a balloon store and a candy store to make money: In the balloon store, Prof. Pang can buy k_{ab} balloons for the price of 1 Au or buy k_{cb} balloons for the price of 1 candy. In the candy store, Prof. Pang can buy k_{ac} candies for the price of 1 Au or buy k_{bc} candies for the price of 1 balloon. Prof. Pang can also sell one balloon and get k_{ba} Aus. He can sell one candy and get k_{ca} Aus. The only constraint to him is that there are only n_b balloons in the balloon store and only n_c candies in the candy store. He can buy balloons and candies only when supplies last. Even if he sells some of his balloons or candies, the number of balloons and candies in the stores will not increase.

Each of the six transactions can be performed in any order for any times (0 or more) but they are not separable (for example, Prof. Pang can not buy $k_{ab}/2$ balloons for the price of $1/2$ Au).

Please find out how many Aus he can make at most.

Input

The first line contains a single integer T ($1 \le T \le 1000$) denoting the number of test cases.

Each of the next T lines contains eight integers n_b, n_c, k_{ab}, k_{ba}, k_{ac}, k_{ca}, k_{bc}, k_{cb} ($1 \le n_b, n_c \le 10^9$, $1 \le k_{ab}, k_{ba}, k_{ac}, k_{ca}, k_{bc}, k_{cb} \le 100$) separated by single spaces.

Output

For each test case, print one line containing the answer.

Example

standard input	standard output
`6`	`7`
`2 2 2 2 2 2 2 2`	`355`
`78 74 5 3 10 2 4 7`	`239`
`31 75 3 6 6 1 8 4`	`571`
`91 86 4 2 9 5 8 5`	`637`
`48 89 3 9 2 3 5 7`	`109`
`13 25 5 7 6 1 2 4`	

Note

In the first example, Prof. Pang buys 2 balloons with 1 Au and then sells 2 balloons and gets 4 Aus. Then he buys 2 candies with 1 Au, sells 2 candies and gets 4 Aus.

Problem I. Plants vs Zombies

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Prof. Pang is playing *Plants vs Zombies*.

Imagine that the game is played on a number axis. The following are the elements in the game:

- n zombies. The i-th zombie appears at 0 on the number axis at time t_i with health point h_i. The zombies have the same moving speed V and they all move to the right.
- m spikeweeds. The i-th spikeweed is of position p_i and attack power a_i.
- One peashooter at the position of 10^{100}. It shoots K peas of attack power D every second.

Every second in the game is processed as follows:

1. When the x-th second begins, the zombies whose t_is equal x appear at position 0.
2. After that, for each appeared and alive zombie u, it will suffer from the spikeweeds whose positions are in $(P_u, P_u + V]$ where P_u is the current position of the u-th zombie. So its health point will be decreased by $\sum\limits_{1\le i\le m, P_u<p_i\le P_u+V} a_i$. The zombie dies if its health point is no more than zero. Otherwise, it is still alive and its position will be increased by V.
3. When the x-th second ends, the peashooter shoots K peas in a row. For each pea, it will hit the zombie that is alive and of the maximum position currently. If there are multiple zombies of the maximum position, the pea hits the one of the minimum index. The health point of the zombie being hit decreases by D. This zombie dies if its health point is decreased to some value no more than 0. The peas are processed one by one, not simultaneously. (For example, if a zombie is killed by the first pea, the second pea cannot hit it since it dies before the second pea is shot.) If no alive zombie exists, the remain peas will hit nothing.

Prof. Pang wants to know the death time (in seconds) of all the n zombies.

Input

The first line contains five integers n, m, V, K, D ($1 \le n, m \le 10^5, 1 \le V, K, D \le 10^9$) separated by single spaces.

Each of the following n lines contains two integers t_i, h_i ($1 \le t_i, h_i \le 10^9$) separated by a single space.

Each of the following m lines contains two integers p_i, a_i ($1 \le p_i, a_i \le 10^9$) separated by a single space.

Output

Output one line containing n integers, where the i-th integer denotes the death time (in seconds) of the i-th zombie.

Example

standard input	standard output

`3 2 1 2 2`
`1 11`
`2 8`
`1 1`
`1 2`
`2 4`
	`2 3 1`

Note

During the first second:

- The first zombie appears and then moves to position 1. It suffers 6 damage points (2 from the first spikeweed, 4 from the two peas).
- The third zombie appears and then moves to position 1. It suffers 2 damage points (from the first spikeweed) and dies (since its health point becomes -1).

During the second second:

- The first zombie moves to position 2 and suffers 6 damage points (4 from the second spikeweed, 2 from the first pea) and dies (since its health point becomes -1).
- The second zombie appears and then moves to position 1. It suffers 4 damage points (2 from the first spikeweed, 2 from the second pea).

During the third second:

- The second zombie moves to position 2, suffers 4 damage points (4 from the second spikeweed) and dies (since its health point becomes 0).
- The peas hit no zombie during this second.

So the death times of the zombies are 2, 3, 1, respectively.

Problem J. Circle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Prof. Pang does research on the minimum covering circle problem. He does not like random algorithms so he decides to find an efficient deterministic one. He starts with the classical idea of binary search. In each iteration of the binary search, the following problem needs to be solved:

Given the radius r of a circle and a convex hull C, let S be defined as

$$S = \{p \mid \text{the circle with center } p \text{ and radius } r \text{ covers } C\}.$$

Find the area of S.

Input

The first line contains a single positive integer T denoting the number of test cases.

For each test case, the first line contains two integers n and r ($1 \le n \le 1000$, $1 \le r \le 30000$) separated by a single space denoting the number of vertices of the convex hull and the radius. If $n = 1$, the convex hull contains only 1 point. If $n = 2$, the convex hull is a line segment.

Each of the following n lines contains two integers x, y ($-10000 \le x, y \le 10000$) separated by a single space denoting a vertex at (x, y). It is guaranteed that no two vertices coincide and no three vertices are collinear. Vertices are listed in counter-clockwise order.

It is guaranteed that the sum of n over all test cases does not exceed 200000.

Output

Output a single decimal indicating the answer. Your answer will be considered correct if the absolute or relative error is no more than 10^{-6}.

Example

standard input	standard output

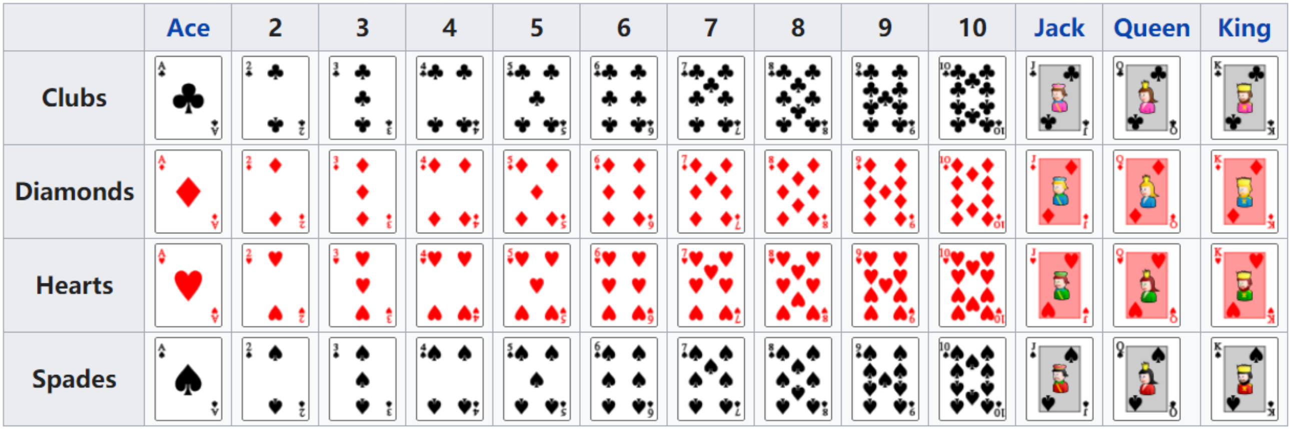
3
4 1
0 0
1 0
1 1
0 1
4 1
0 0
1 1
0 2
-1 1
4 100
0 0
1 0
1 1
0 1
	0.315146743628
	0
	31016.928202570849

Problem K. Allin

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Texas hold ’em (also known as **Texas holdem**, **hold ’em**, and **holdem**) is one of the most popular variants of the card game of poker. **Please read the following rules as they may be different from the regular rules.** **Two** cards, known as **hole cards**, are dealt **face-down** to each player. Each player only knows his own hole cards. And then **five community cards** are dealt in three stages **face-up**. The stages consist of a series of three cards (“the flop”), later an additional single card (“the turn” or “fourth street”), and a final card (“the river” or “fifth street”). All players know the face-up cards that are already dealt. All cards are drawn from a standard 52-card deck. A standard 52-card deck comprises 13 ranks in each of the four French suits: clubs (♣), diamonds (♢), hearts (♡) and spades (♠). Each suit includes an Ace (A), a King (K), Queen (Q) and Jack (J), each depicted alongside a symbol of its suit; and numerals or pip cards from the Deuce (Two) to the Ten, with each card depicting that many symbols (pips) of its suit. **No card can be drawn more than once.**

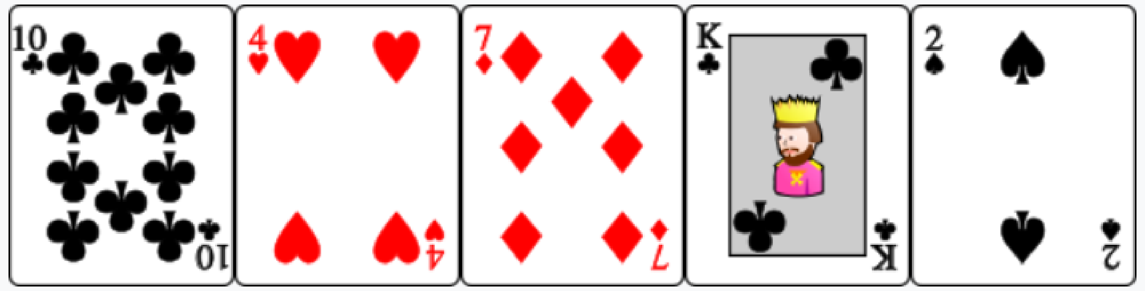
Example set of 52 playing cards; 13 of each suit: clubs, diamonds, hearts, and spades



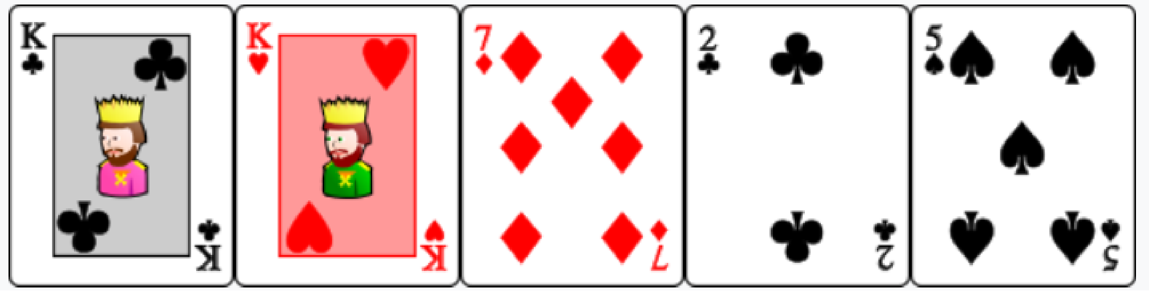
Individual cards are ranked as follows (high-to-low): A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2. **Each player seeks the best five-card poker hand from any combination of the seven cards – the five community cards and his two hole cards.**

The following table shows the possible five-card poker hand types in **increasing order** of their values. Each type has a specific ordering of the five cards that is described below. **The following part is describing how to compare two hands, which is the same as the regular rule.**

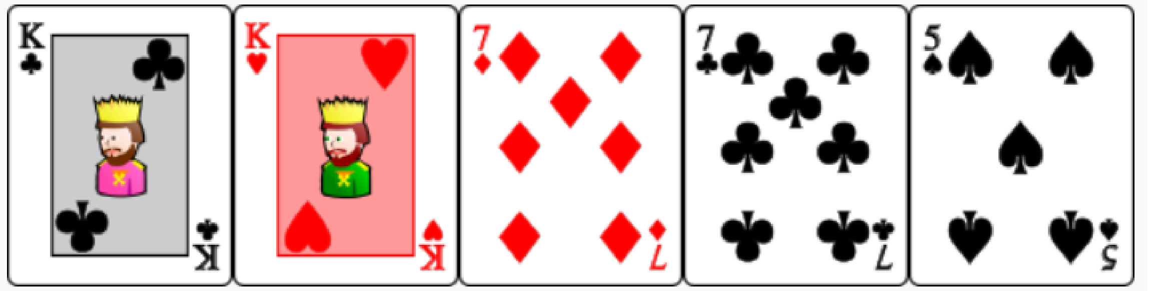
- **Highcard**: Simple value of the card. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 > a_2 > a_3 > a_4 > a_5$. ($a_i$ represents the rank of i-th card.)



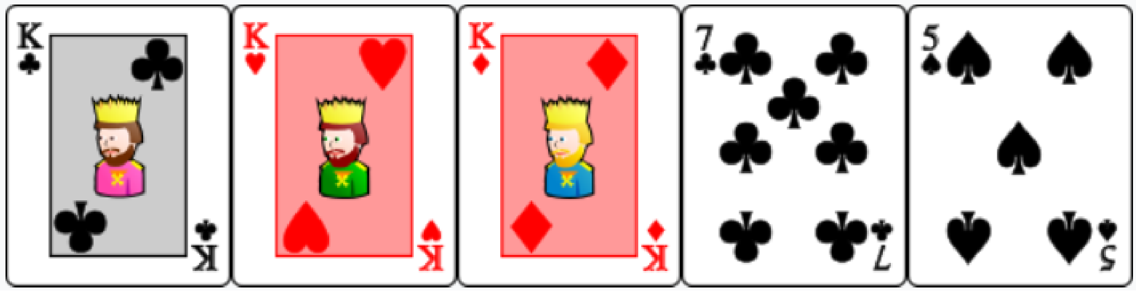
- **Pair**: Two cards with the same value. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 = a_2$, $a_3 > a_4 > a_5$, $a_1 \neq a_3$, $a_1 \neq a_4$, $a_1 \neq a_5$.



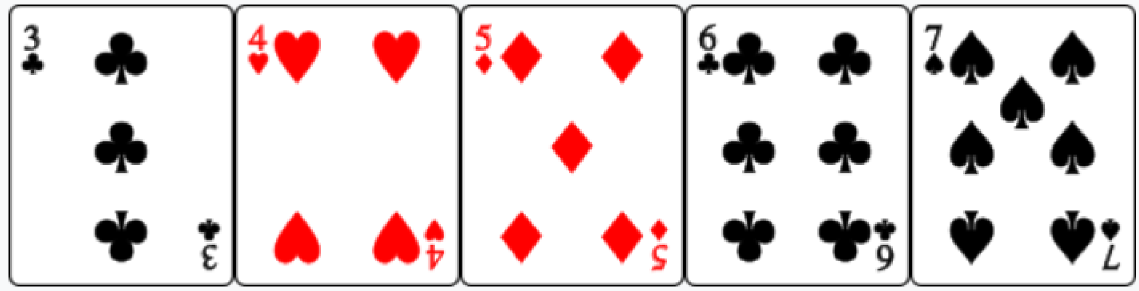
- **Two pairs**: Two times two cards with the same value. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 = a_2$, $a_3 = a_4$, $a_1 > a_3$, $a_1 \neq a_5$, $a_3 \neq a_5$.



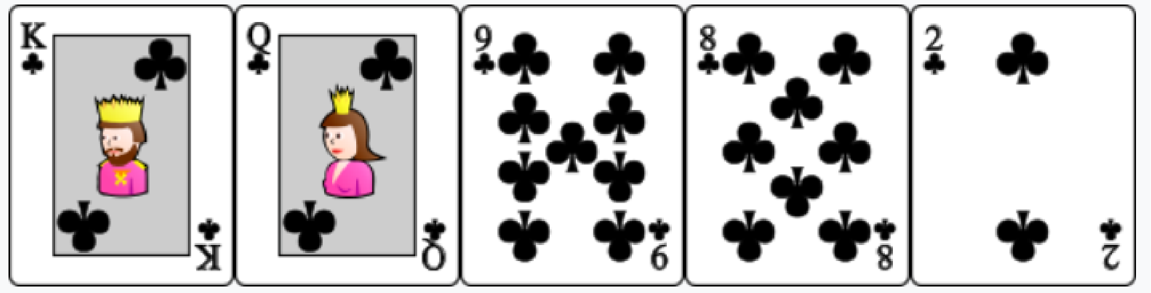
- **Three of a kind**: Three cards with the same value. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 = a_2 = a_3$,$a_4 > a_5$, $a_1 \neq a_4$, $a_1 \neq a_5$.



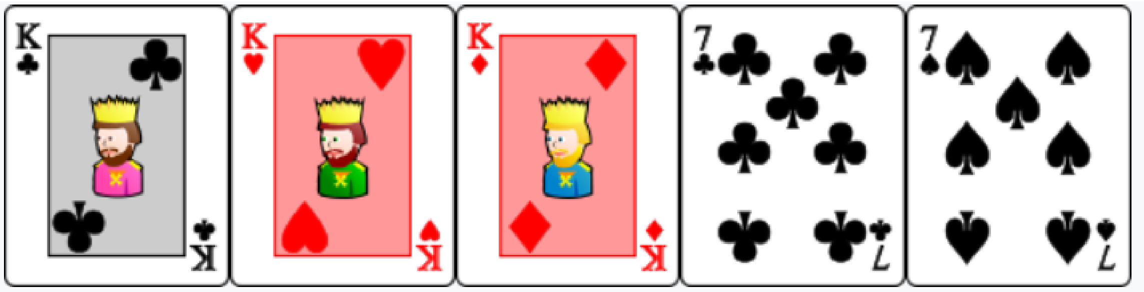
- **Straight**: Sequence of 5 cards in increasing value. The cards are ordered as $a_1a_2a_3a_4a_5$ such that a_i is exactly one rank above a_{i+1} for all $1 \leq i \leq 4$. **Specially, if a_5 is Ace, a_4 can be 2. In this case, Ace is considered one rank below 2.**



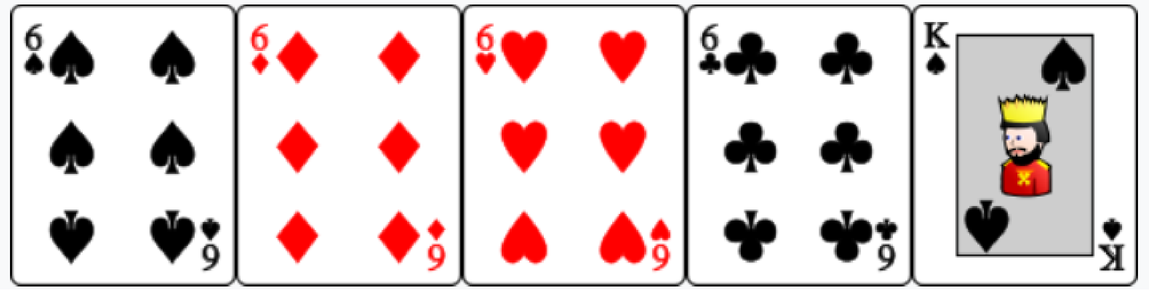
- **Flush**: 5 cards of the same suit. The cards are ordered as $a_1a_2a_3a_4a_5$ such that all the five cards have the same suit and $a_1 > a_2 > a_3 > a_4 > a_5$.



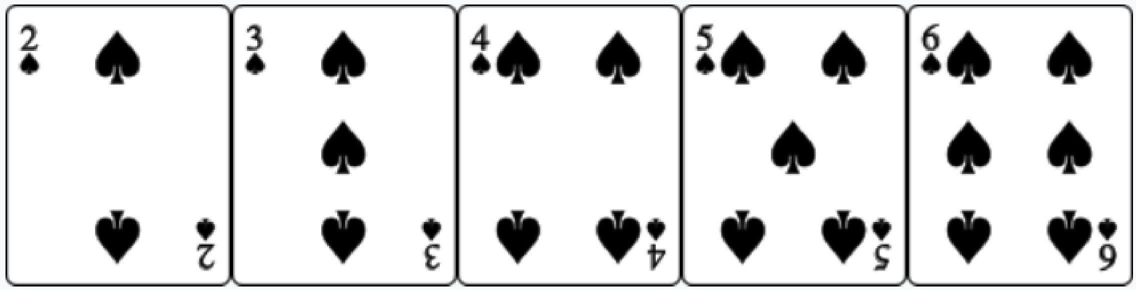
- **Full house**: Combination of three of a kind and a pair. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 = a_2 = a_3$, $a_4 = a_5$.



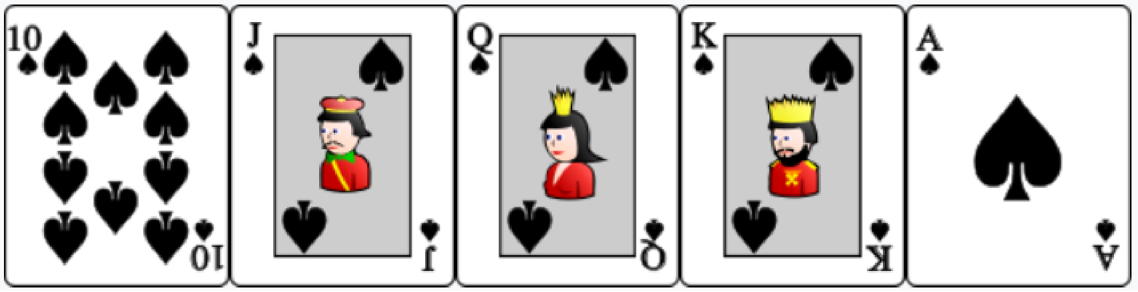
- **Four of a kind**: Four cards of the same value. The cards are ordered as $a_1a_2a_3a_4a_5$ such that $a_1 = a_2 = a_3 = a_4$.



- **Straight flush**: Straight of the same suit. The cards are ordered as $a_1a_2a_3a_4a_5$ such that all the five cards have the same suit and that a_i is exactly one rank above a_{i+1} for all $1 \le i \le 4$. **Specially, if a_5 is Ace, a_4 can be 2. In this case, Ace is considered one rank below 2.**



- **Royal flush**: Straight flush from Ten to Ace. The cards are ordered as $a_1a_2a_3a_4a_5$ such that a_1, a_2, a_3, a_4, a_5 are Ace, King, Queen, Jack, Ten of the same suit.



To compare two hands, first, we will compare the type of two hands. For example, one hand is **Four of a kind**, the other hand is **Full house**, **Four of a kind** always win **Full house**.

If the types of two hands are the same, we compare the ranks of the cards. We will order the card as described above, and compare them one by one. Firstly, we will compare the first card. If a hand's first card has a higher rank, it wins. If the first cards of the two hands have the same rank, we will compare the second card, and so on. If the cards have the same rank in every position, no one wins. The suit of cards never matters. For example, ♣ 5, ♢ 5, ♡ 5, ♠ 2, ♣ 2 can win ♢ 3, ♠ 3, ♡ 3, ♢ A, ♡ A. Since they are both **Full house**, and we will compare the ranks of the three cards of a kind at first.

Consider the case that the hole cards of Alice are ♣ A, ♢ 4 and the hole cards of Bob are ♡ 2, ♠ 3. The community cards are ♠ A, ♡ 4, ♠ 5, ♣ Q, ♡ Q. The best hand of Alice (five cards among her hole cards and the community hards) is ♣ A, ♠ A, ♣ Q, ♡ Q, ♠ 5, which is **Two pairs**. The best hand of Bob is ♠ 5, ♡ 4, ♠ 3, ♡ 2, ♠ A, which is **Straight**. Thus, Bob wins.

Players have betting options to check, call, raise, or fold. **In this problem, we do not care about the meanings of these bets.** Rounds of betting take place before the flop is dealt and after each subsequent deal. The player who has the best hand and has not folded by the end of all betting rounds wins all of the money bet for the hand, known as the pot. In certain situations, a “split-pot” or “tie” can occur when two players have hands of equivalent value. This is also called a “chop-pot”. **In this problem, we assume the two players never fold.** So the player with the best five-card poker hand from any combination of the seven cards wins. If the two players have hands of equal values, no one wins.

To simplify the statement, we do not introduce the detailed rules here.

Daddy Dream is a world-famous **Texas hold ’em** player. As a strong challenger, Wolf Chicken wants to beat Daddy Dream. Wolf Chicken plays first after “the flop” (three cards are dealt face-up). Both players know the three face-up cards and each player knows his own two hole cards. Wolf Chicken will choose to allin if and only if he will certainly win whatever the “the turn”, “the river” (the remaining two community cards that have not been revealed) and Daddy Dream’s hole cards are. Otherwise, Wolf Chicken will choose to check.

Given Wolf Chicken’s two hole cards and the three flop cards, help him to determine whether he can allin.

Input

The first line contains a single integer T ($1 \le T \le 100000$) denoting the number of test cases.

For each test case, there is one line containing five strings h_1, h_2, c_1, c_2, c_3 separated by single spaces denoting the first hole card, the second hole card, the first community card, the second community card and the third community card.

For each card, the first character of its corresponding string denotes its rank. (Possible ranks are ‘2’ ,‘3’,‘4’,‘5’,‘6’,‘7’,‘8’,‘9’,‘T’,‘J’,‘Q’,‘K’,‘A’. ‘T’ denotes 10.) The second character denotes its suit. ‘C’ denotes clubs. ‘D’ denotes diamonds. ‘H’ denotes hearts. ‘S’ denotes spades.

It is guaranteed that each card appears at most once in one test case.

Output

For each test case, print one line. Print “`allin`” if Wolf Chicken will certainly win. Otherwise, print “`check`”.

Example

standard input	standard output

`2`
`AC KC QC JC TC`
`AC TD 8S 5H 2C`
	`allin`
	`check`

Problem L. Square

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Father Study loves math very much.

Given a sequence of integers $a_1, a_2, ..., a_n$, Father Study wants to calculate another sequence of integers $t_1, t_2, ..., t_n$ satisifing

- For each i $(1 \le i \le n)$, $t_i > 0$.
- For each i $(1 \le i < n)$, $a_i \times t_i \times a_{i+1} \times t_{i+1}$ is a square number. (In mathematics, a square number or perfect square is an integer that is the square of an integer, in other words, it is the product of some integer with itself.)
- $\prod_{i=1}^{n} t_i$ is minimized.

Please help Father Study to calculate the answer — the minimum value of $\prod_{i=1}^{n} t_i$. Because the answer is too large, please output the answer modulo 1000000007.

Input

The first line contains a single integer n $(1 \le n \le 100000)$.

The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 1000000)$ separated by single spaces.

Output

Output one integer – the answer modulo 1000000007.

Example

standard input	standard output
3 2 3 6	6

Problem M. Fillomino

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Prof. Pang is the king of Pangland. Pangland is a board with size $n \times m$. The cell at the i-th row and the j-th column is denoted as cell (i, j) for all $1 \le i \le n, 1 \le j \le m$. If two cells share an edge, they are connected. The board is **toroidal**, that is, cell $(1, y)$ is also connected to (n, y) and $(x, 1)$ is also connected to (x, m) for all $1 \le x \le n, 1 \le y \le m$.

Prof. Pang has three sons. We call them the first son, the second son and the third son. Each of them lives in a cell in Pangland. The i-th son lives in cell (x_i, y_i). No two sons live in the same cell. Prof. Pang wants to distribute the cells in Pangland to his sons such that

- Each cell belongs to exactly one son.
- There are cnt_i cells that belong to the i-th son for all $1 \le i \le 3$.
- The cells that belong to the i-th son are connected for all $1 \le i \le 3$.
- The cell that the i-th son lives in must belong to the i-th son himself for all $1 \le i \le 3$.

Please help Prof. Pang to find a solution if possible.

Input

The first line contains a single integer T $(1 \le T \le 10^5)$ denoting the number of test cases.

For each test case, the first line contains two integers n, m $(3 \le n, m \le 500)$ separated by a single space.

The next line contains three positive integers cnt_1, cnt_2, cnt_3 $(cnt_1 + cnt_2 + cnt_3 = nm)$ separated by single spaces.

The i-th line of the next 3 lines contains two integers x_i, y_i $(1 \le x_i \le n, 1 \le y_i \le m)$ separated by a single space.

It is guaranteed that (x_1, y_1), (x_2, y_2), (x_3, y_3) are distinct.

It is guaranteed that the sum of nm over all test cases is no more than 10^6.

Output

For each test case, if there is no solution, output “`-1`” in one line. Otherwise, output n lines. Each line should contain m characters. The j-th character in the i-th line should be ‘`A`’ if cell (i, j) belongs to the first son, ‘`B`’ if cell (i, j) belongs to the second son and ‘`C`’ if cell (i, j) belongs to the third son. Cell (x_i, y_i) must belong to the i-th son for all $1 \le i \le 3$. The cells that belong to the i-th son must be connected for all $1 \le i \le 3$.

Example

standard input	standard output

2
3 3
1 3 5
1 1
2 2
3 3
4 4
5 5 6
2 2
2 3
3 3
	ABB
	CBC
	CCC
	BABB
	BABC
	CACC
	AACC