A Family of Bipartite Graphs (BIPFAMIL)

In this problem, we study about Range Graphs - a special family of bipartite graphs. A bipartite graph with n vertices on the left part (numbered from 1 to n) and m on the right part (numbered from $n+1$ to $n+m$), is said to be a Range Graph, if each vertex on the left satisfies the following property - the numbers of its set of neighbors should form a consecutive range/segment on the right.

For example, suppose n = 10 and m = 13. Then the vertices on the left are {1, 2, …, 10} and the ones on the right are {11, 12, …, 23}. Now, suppose the neighbors of vertex 7 were {12, 14, 15, 19}, then this would not be a Range Graph, because they aren't consecutive numbers. So for it to be a Range Graph, the neighbors could potentially be, say, {12, 13, 14, 15}.

Your task is to find the total number of **connected** Range Graphs which have n vertices on left and m on the right. Output your answer modulo 10^9+7.

Note that two Range Graphs are considered to be different if there are two vertices i and j such that one of the graphs has an edge between them, but the other graph doesn't. In other words, the edge set should be different. A graph is said to be connected if every vertex is reachable from every other vertex through some sequence of edges.

Note: The source limit (ie. the size of your program file) for this problem is lower than usual. It is 10 KB.

Input

- The first line of the input contains an integer T denoting the number of test cases. The description of the test cases follows.
- The only line of each test case contains two space-separated integers n, m.

Output:

For each test case, output an integer corresponding to the answer of the problem.

Constraints

- $1 \le T \le 10^5$
- $1 \le n, m \le 2500$
- $1 \le n * m \le 2500$

Sample Input:

```
2
1 2
2 2
```

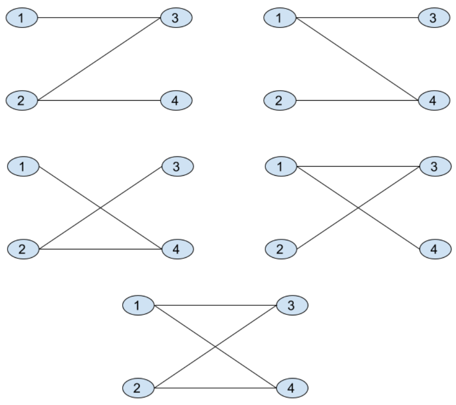
Sample Output:

```
1
5
```

Explanation

Example 1: There is only one possible connected Range Graph with one vertex on the left and two vertices on the right. It will have an edge between vertices 1 and 2, and also between vertices 1 and 3.

Example 2 Here are the five possible connected Range Graphs with 2 vertices on left and right:



The Prestige (PRESTIGE)

Robert Angier and his assistant are back in town for another magic trick (nothing as fancy as “The Transported Man”, but it’s still good).

The Setup:

The audience is shown two separate deck of cards. Both have N cards in them, and each deck is arranged in a stack.

Each card of the first deck has the number **1** written on one face, and **-1** on the other face. Initially, the deck is arranged so that all the cards have **1** on upper face, and **-1** on lower.

Each card of the second deck has integers written on both faces. These integers do not have to be the same. The order in which these cards are stacked is given to you.

The Performance:

Robert does one of the following three actions every minute:

- Type 1: He asks some random person in the audience to pick two numbers L and R. Then he picks the cards at positions L to R from the top of the second deck, flips them as a whole, and reinserts them at the same position from where he extracted them. This flipping process has been described below in the 'Flipping' section in more detail. This action is denoted by $1\ L\ R$.
- Type 2: He asks his assistant to pick a number K. Then he picks the first K cards from the top of the first deck, flips them as a whole, and places them back on the same deck. This flipping process has been described below in the 'Flipping' section in more detail. This action is denoted by $2\ K$.
- Type 3: He asks some random person in the audience to pick four numbers A, B, C and D such that (B - A) == (D - C). This action is denoted by $3\ A\ B\ C\ D$. Then he proceeds to perform The Prestige.

The Prestige:

Robert instantly reports the value of
$(first[C] * second[A]) + (first[C+1] * second[A+1]) + \cdots + (first[D] * second[B])$

where $first[i]$ and $second[i]$ are values on the upper faces of i-th card of the first and second decks, respectively.

The Trick:

When asked to pick a number K, the assistant always picks a value at least as large as the one he previously picked (The first time he is asked to pick, he picks some integer $\geq$ 0).

Flipping:

Suppose you have a deck with 6 cards, with the card at index 1 on the top of the stack, and the card with index 6 at the bottom of the stack. Let the values written on each of the cards be as follows:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |      a     |      b     |
|   2   |      c     |      d     |
|   3   |      e     |      f     |
|   4   |      g     |      h     |
|   5   |      i     |      j     |
|   6   |      k     |      l     |
-----------------------------------
```

After you flip from positions 2 to 5 (ie. doing the operation $1\ 2\ 5$), it will become

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |      a     |      b     |
|   2   |      j     |      i     |
|   3   |      h     |      g     |
|   4   |      f     |      e     |
|   5   |      d     |      c     |
|   6   |      k     |      l     |
-----------------------------------
```

In other words, when you perform the action $1\ L\ R$, you 'flip' the cards from L to R, which means that you reverse the order of these cards, and also change the orientation of each of these cards. That is, the old upper face now becomes the new lower face, and vice versa. The action $2\ K$ simply refers to flipping the first K cards of the first deck.

Robert however has become distracted with Borden's more impressive performances, and hence you have to help him perform this trick. Given the sequence of actions that is to be done, output the answer for every action of the third type.

Input:

- The first line contains N and M, the number of cards in each deck, and the total number of actions Robert does.
- The second line contains N integers, representing the values on the upper face of the second deck, from the top of the deck to bottom.
- The third line contains N integers, representing the values on the lower face of the second deck, from the top of the deck to bottom.
- The following M lines contain descriptions of the actions. Each action can be represented as one of the following (corresponding to the actions described in The Performance section):
  - $1\ L\ R$
  - $2\ K$
  - $3\ A\ B\ C\ D$

Output:

- For every action of type 3, output its answer in a new line.

Constraints:

- $1 \leq N \leq 200000$
- $1 \leq M \leq 200000$
- $-10^9 \leq$ integers on the cards $\leq 10^9$
- $1 \leq L \leq R \leq N$
- $0 \leq K \leq N$
- $1 \leq A \leq B \leq N$
- $1 \leq C \leq D \leq N$
- The values of K in the actions of type 2 will be in non-decreasing order.

Sample Input:

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

Sample Output:

```
8
16
10
21
```

Sample Explanation:

- Initially, the first deck is as follows:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |      1     |     -1     |
|   2   |      1     |     -1     |
|   3   |      1     |     -1     |
|   4   |      1     |     -1     |
|   5   |      1     |     -1     |
|   6   |      1     |     -1     |
|   7   |      1     |     -1     |
-----------------------------------
```

And the second deck is as follows:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |      3     |      2     |
|   2   |      5     |      8     |
|   3   |      0     |      4     |
|   4   |      2     |      4     |
|   5   |      5     |      0     |
|   6   |      7     |      1     |
|   7   |      4     |      6     |
-----------------------------------
```

- In the first action, you want to find $(first[3] * second[1]) + (first[4] * second[2]) + (first[5] * second[3])$
  $= (1 * 3) + (1 * 5) + (1 * 0) = 8.$

  Hence the first output is 8.
- After the second action, the first deck becomes:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |     -1     |      1     |
|   2   |      1     |     -1     |
|   3   |      1     |     -1     |
|   4   |      1     |     -1     |
|   5   |      1     |     -1     |
|   6   |      1     |     -1     |
|   7   |      1     |     -1     |
-----------------------------------
```

- After the third action, the second deck becomes:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |      3     |      2     |
|   2   |      1     |      7     |
|   3   |      0     |      5     |
|   4   |      4     |      2     |
|   5   |      4     |      0     |
|   6   |      8     |      5     |
|   7   |      4     |      6     |
-----------------------------------
```

- After the fourth action, the second deck becomes:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |     3      |     2      |
|   2   |     6      |     4      |
|   3   |     5      |     8      |
|   4   |     0      |     4      |
|   5   |     2      |     4      |
|   6   |     5      |     0      |
|   7   |     7      |     1      |
-----------------------------------
```

- After the fifth action, the first deck becomes:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |    -1      |     1      |
|   2   |    -1      |     1      |
|   3   |     1      |    -1      |
|   4   |     1      |    -1      |
|   5   |     1      |    -1      |
|   6   |     1      |    -1      |
|   7   |     1      |    -1      |
-----------------------------------
```

- In the sixth action, you want to find
  $(first[3] * second[1]) + (first[4] * second[2]) + (first[5] * second[3])$
  $+(first[6] * second[4]) + (first[7] * second[5])$
  $= (1 * 3) + (1 * 6) + (1 * 5) + (1 * 0) + (1 * 2) = 16$

  Hence the second output is 16.
- In the seventh action, you want to find
  $(first[1] * second[1]) + (first[2] * second[2]) + (first[3] * second[3])$
  $+(first[4] * second[4]) + (first[5] * second[5]) + (first[6] * second[6])$
  $+(first[7] * second[7])$
  $= (-1 * 3) + (-1 * 6) + (1 * 5) + (1 * 0) + (1 * 2) + (1 * 5) + (1 * 7) = 10$

  Hence the third output is 10.
- After the eighth action, the second deck becomes:

```
-----------------------------------
| Index | Upper Face | Lower Face |
|-------|------------|------------|
|   1   |     0      |     5      |
|   2   |     4      |     2      |
|   3   |     4      |     0      |
|   4   |     8      |     5      |
|   5   |     4      |     6      |
|   6   |     2      |     3      |
|   7   |     7      |     1      |
-----------------------------------
```

- In the ninth action, you want to find
  $(first[1] * second[1]) + (first[2] * second[2]) + (first[3] * second[3])$
  $+(first[4] * second[4]) + (first[5] * second[5]) + (first[6] * second[6])$
  $+(first[7] * second[7])$
  $= (-1 * 0) + (-1 * 4) + (1 * 4) + (1 * 8) + (1 * 4) + (1 * 2) + (1 * 7) = 21$

  Hence the fourth output is 21.

Reduction Game (REDCGAME)

You are playing a game called the Reduction Game. This game is played on an array a with n integers. You also have another integer k with you. The following operation will be made repeatedly on this array:

- Choose any two elements a_i, a_j such that $i \neq j$, and $min(a_i, a_j) > k$. Decrease both a_i and a_j by one.

This operation should be applied to the array if it's possible to apply it. But if there are multiple valid choices of a_i and a_j, you can choose which ones to do first and so on. The aim is to apply the operations in such a way, so as to maximize the sum of the elements in the array at the end. Your task is to find the maximum possible sum of the final array.

Input

- The first line of the input contains an integer T denoting the number of test cases. The description of the test cases follows.
- The first line of each test case contains two integers n, k denoting the number of elements in the array a.
- The second line contains n space-separated integers denoting the array a.

Output:

For each test case, output an integer corresponding to the maximum possible sum of the remaining array in a new line.

Constraints

- $1 \leq T \leq 5000$
- $1 \leq n \leq 50$
- $1 \leq a_i, k \leq 50000$

Sample Input:

```
3
2 1
1 2
2 1
2 2
3 1
2 3 2
```

Sample Output:

```
3
2
5
```

Explanation

Example 1: The initial array is [1, 2]. In this case, there are no two elements > 1. So, no operations can be made, and this is the only final array possible. And the sum of this is 3, and hence that is the answer.

Example 2: The initial array is [2, 2]. You can decrease 1 from both the elements and get the array to [1, 1]. This is the only valid pair available, so you are forced to reduce this pair. After doing so, you can't make any more operations. So, [1, 1] is the only final array achievable, and it's sum is is 2.

Example 3: The initial array is [2, 3, 2]. You could pick the first two elements and reduce them, to get the array [1, 2, 2]. Now, you have to choose the last two elements and reduce them, to get [1, 1, 1]. This is one possible final array, whose sum is 3.

Instead, you could initially have chosen the first and third element and reduced them to get [1, 3, 1]. After this, there is no operation possible, and so this is another final array achievable. Its sum is 5, and there is nothing better than this. Hence the answer is 5.

A Game of Robots (ROBOGAME)

You've built some robots and placed them on an one dimensional grid with n cells. Their arrangement is given by a string s of length n. Each character of the string is either '.' or a digit in the range '0' to '9'. A '.' represents that there is no robot at that cell initially. A digit represents a robot initially in that cell. Specifically, the digit x denotes that the range of the robot is from x cells to the left of its starting point to x cells to the right of its starting point.

For example, suppose the 7th character of the string is 3, then that means that there is a robot starting from the 7th cell, and its range is from the 4th cell (7 - 3 = 4) to the 10th cell (7 + 3 = 10) (both end points inclusive). The robots can move only within their range, and even if their range allows it, they cannot move out of the grid.

You want to play a game with these robots. Before starting the game, you can give each robot a starting direction (either left or right). When the robot is initialized with a direction, it will move in that direction until it can (ie. it can't go past its range, and neither can it go outside the grid) and will reverse its direction and go as far as possible in that direction, and then reverse, and so. It will keep going like this forever. Assume that the change of direction happens instantaneously.

But the catch is that each of the robots can start their journey at any time. They don't all have to start at the same second. It won't stop once it starts moving though. And they all move at the same speed of one cell per second, once they start.

The robots have gained consciousness, and have begun questioning their purpose in life. So given a chance, they will collide with each other and end their misery. They can coordinate with each other as well and decide when they should start their journeys. Two robots are said to have collided if they are at the same cell at the same moment.

You are wondering whether it is possible to give the robots the initial directions in such a way that no robots collide with each other (ie. they'll all be safe), or if no matter what initial directions you give, some of them will end up colliding with each other (ie. unsafe).

Input

- The first line of the input contains an integer T denoting the number of test cases. The description of the test cases follows.
- The only line of each test case contains a string s.

Output

- For each test case, output a single line containing "safe" or "unsafe" (without quotes).

Constraints

- $1 \leq T \leq 3 * 10^4$
- $1 \leq$ length of $s \leq 50$
- $s[i]$ will be one of the following characters {'.', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'}

Example

Input

```
4
....
.2.....
.2...2..
1.1.1.
```

Output

```
safe
safe
unsafe
unsafe
```

Explanation

Example 1: No robots. Everything is safe.

Example 2: Only one robot. Everything safe. To give an example of its movement, suppose you give it the starting direction as left and suppose it decides to start moving at time = 5 seconds. Then till t = 5, it is at cell 2 (its starting position, 1-based indexing). At t = 6, it is at cell 1. At t = 7, it is at cell 2, at t = 8, it will be at cell 3, at t = 9 it will be at cell 4 and at t = 10 it will be at cell 3. And so on.

Example 3: No matter what initial directions you give, the two robots can coordinate and start their journeys such that they collide at the cell 4.

Word Grid (WORDGRID)

You are given a set of n words, each of length 4. All characters are uppercase English alphabets. You have to construct a 4x4 grid of letters such that each of the n words appears atleast once.

A word is said to appear in a grid if the word appears in a row or column in any of the two directions (ie. if it is a column, it can either be top to bottom, or bottom to top. And if it is a row, it can either be left to right, or right to left).

If there are many such grids, return the lexicographically smallest grid.

A grid A is said to be lexicographically smaller than grid B if A has the smaller letter in the first unequal square (in row major order from top to bottom, left to right). That is, you first check if there is any unequality in the first row, and if not in the second row, etc. And in each row, you check from left to right.

If there are no such grids, print the word "grid's not possible" in the form of 4x4 grid without the punctuations or spaces i.e.

```
grid
snot
poss
ible
```

Input:

- First line will contain T, number of testcases. Then the testcases follow.
- Each testcase will start with n, the number of words in the input.
- Each of the next n lines will contain a word of length 4.

Output:

For each testcase, output the lexicographically smallest 4x4 grid containing all the given words or if that is not possible, print the not possible grid given in the statement.

Output a blank line after each test case.

Constraints

- $1 \leq T \leq 50$
- $1 \leq n \leq 32$
- All the characters in the n words are uppercase English alphabets. ie. 'A' to 'Z'.

Sample Input:

```
2
6
CODE
TRAP
ARCS
CART
ROAD
PART
5
ABCD
EFGH
IJKL
MNOP
QRST
```

Sample Output:

```
CART
OROR
DCAA
ESDP

grid
snot
poss
ible
```

Explanation:

Example 1:

Case 1

- CODE occurs in the first column (top to bottom)
- TRAP in the fourth column (top to bottom)
- ARCS in the second column (top to bottom)
- CART in the first row (left to right)
- ROAD in the third column (top to bottom)
- PART in the fourth column (bottom to top)

Case 2

There is no 4x4 grid that can have all the given words