Задача 1. Реализовать на языке программирования Си параллельную программу и теста к ней с помощью одной или нескольких технологий

- OpenMP
- MPI
- MPI+OpenMP

для максимально быстрого решения одной задачи из списка:

1. Даны две вещественные матрицы **A** и **B**. Вычислить матрицу **C**=**A**·**B**.
2. Дана вещественная матрица **A** и матрица **У**. Найти решение системы уравнений **A**·**X**=**У**.
3. Дана вещественная треугольная матрица **A** и матрица **У**. Найти решение системы уравнений **A**·**X**=**У**.
4. Даны вещественные или комплексные векторы **x** и **y**, вычислить их скалярное произведение.
5. Дан вещественный или комплексный вектор **x**, найти первый по порядку максимальный по модулю элемент вектора.
6. Дана вещественная матрица **A** и вектор **x**. Найти их произведение.
7. Дана вещественная матрица **A**, нужно транспонировать её.

Предоставить программу и тест к ней в виде исходного кода. Предоставить краткое описание способа их использования, применённые подходы к ускорению и мотивацию по выбору этих подходов к ускорению. Решение более чем одной задачи будет плюсом.

Huawei, Novosibirsk Research Center

Supplementary tasks of the Compiler and Programming Languages Lab

Below you will find three supplementary tasks. Before starting, please read solution guidelines carefully. Thank you!

Solution guidelines

Keep in mind that we do not want to test your skills in a particular programming language or experience with a particular programming system. Do **not** use hacks and tricks, class libraries, and GUI. Make your solution as readable as possible by carefully formatting it and including a **reasonable** amount of comments.

Do not spend time on file handling; standard input and output are sufficient.

A solution must include:

- Your name(s)
- Source code.
- A **representative set of tests** (pay close attention to it!)

The quality of the used algorithms, source code, comments and test suite are most important. And quality is more important than quantity. It's better to prepare, say, one **high quality** solution than to submit low quality solutions for all three problems.

Preferred languages are C, C++, and Java. Scala, Python, Ruby, etc., are okay too.

Objective-C, PHP, and Visual Basic, albeit popular, are not good choices. Do **not** include binaries. We can compile your sources with Oracle JDK, GNU C, or MSVC, so please run your solution through one of those compilers before submitting it.

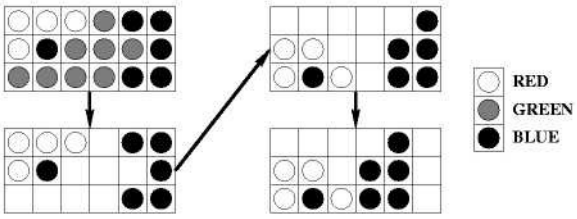
Good luck!

The RGB Game

The game named "RGB" is a single-person game played on a 10 X 15 board. Each square contains a ball colored red (R), green (G), or blue (B). Two balls belong to the same cluster if they have the same color, and one can be reached from another by following balls of the same color in the four directions up, down, left, and right. At each step of the game, the player chooses a ball whose cluster has at least two balls and removes all balls in the cluster from the board. Then, the board is "compressed" in two steps:

1. Shift the remaining balls in each column down to fill the empty spaces. The order of the balls in each column is preserved.
2. If a column becomes empty, shift the remaining columns to the left as far as possible. The order of the columns is preserved.

For example, choosing the ball at the bottom left corner in the sub-board below causes:



The objective of the game is to remove every ball from the board, and the game is over when every ball is removed or when every cluster has only one ball. The scoring of each game is as follows. The player starts with a score of 0. When a cluster of m balls is removed, the player's score increases by $(m - 2)^2$. A bonus of 1000 is given if every ball is removed at the end of the game.

You suspect that a good strategy might be to choose the ball that gives the largest possible cluster at each step, and you want to test this strategy by writing a program to simulate games played using this strategy. If there are two or more balls to choose from, the program should choose the leftmost ball giving the largest cluster. If there is still a tie, it should choose the bottommost ball of these leftmost balls.

Input

You will be given a number of games in the input. The first line of input contains a positive integer giving the number of games to follow. The initial arrangement of the balls of each game is given one row at a time, from top to bottom. Each row contains 15 characters, each of which is one of "R", "G", or "B", specifying the colors of the balls in the row from left to right. A blank line precedes each game.

Output

For each game, print the game number, followed by a new line, followed by information about each move, followed by the final score. Each move should be printed in the format:

Move x at (r,c): removed b balls of color C, got s points.

where x is the move number, r and c are the row number and column number of the chosen ball, respectively. The rows are numbered from 1 to 10 from the bottom, and columns are numbered from 1 to 15 from the left, **b** is the number of balls in the cluster removed. C is one of "R", "G", or "B", indicating the color of the balls removed, **s** is the score for this move. The score does not include the 1000 point bonus if all the balls are removed after the move.

The final score should be reported as follows:

Final score: s, with b balls remaining.

Insert a blank line between the output of each game. Use the plural forms "balls" and "points" even if the corresponding value is 1.

Sample Input

```
3

RGGBBGGRBRRGGBG
RBGRBGRBGRBGRBG
RRRRGBBBRGGRBBB
GGRGBGGBRRGGGBG
GBGGRRRRRBGGRRR
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRGGGGRRRRR
GGGGGGGGGGGGGGG

RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG

RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
```

Sample Output

```
Game 1:
Move 1 at (4,1): removed 32 balls of color B, got 900 points.
Move 2 at (2,1): removed 39 balls of color R, got 1369 points.
Move 3 at (1,1): removed 37 balls of color G, got 1225 points.
Move 4 at (3,4): removed 11 balls of color B, got 81 points.
```

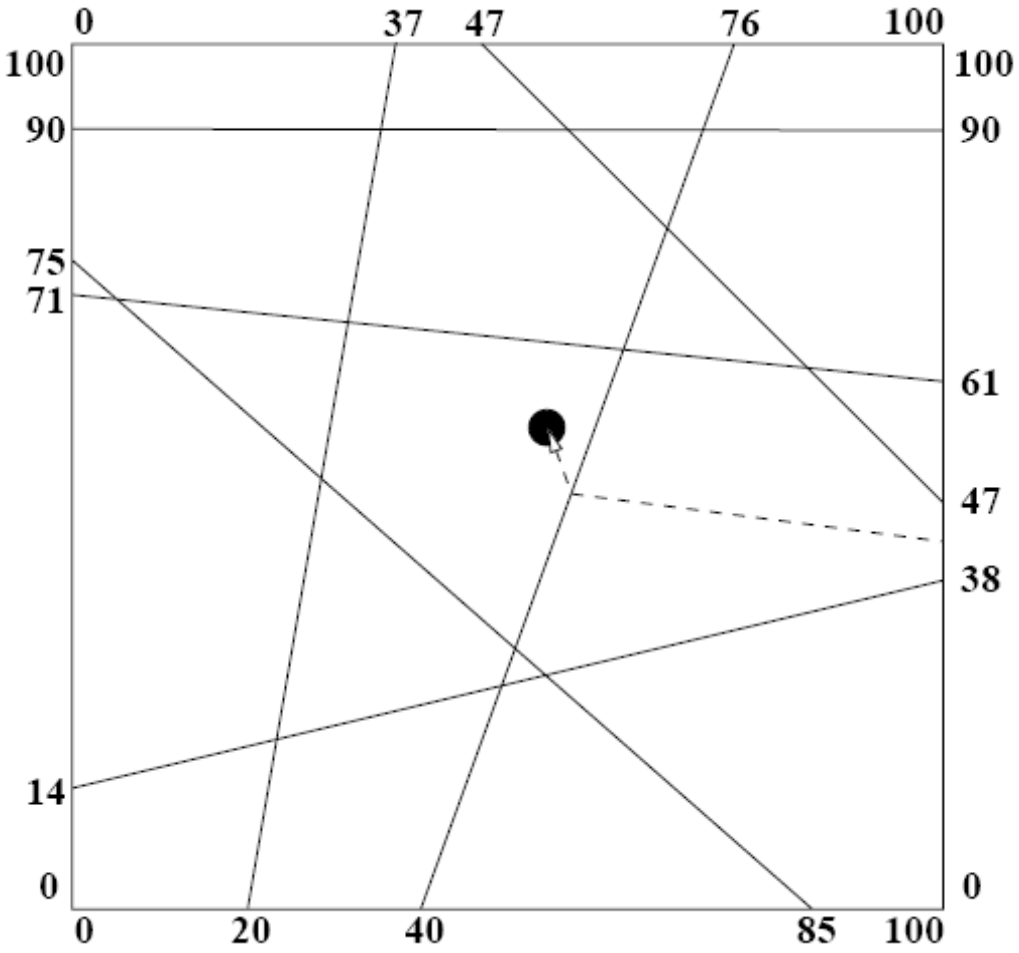
```
Move 5 at (1,1): removed 8 balls of color R, got 36 points.
Move 6 at (2,1): removed 6 balls of color G, got 16 points.
Move 7 at (1,6): removed 6 balls of color B, got 16 points.
Move 8 at (1,2): removed 5 balls of color R, got 9 points.
Move 9 at (1,2): removed 5 balls of color G, got 9 points.
Final score: 3661, with 1 balls remaining.

Game 1:
Move 1 at (1,1): removed 30 balls of color G, got 784 points.
Move 2 at (1,1): removed 30 balls of color R, got 784 points.
Move 3 at (1,1): removed 30 balls of color B, got 784 points.
Move 4 at (1,1): removed 30 balls of color G, got 784 points.
Move 5 at (1,1): removed 30 balls of color R, got 784 points.
Final score: 4920, with 0 balls remaining.
Game 3:
Final score: 0, with 150 balls remaining.
```

Treasure Hunt

Archaeologists from the Antiquities and Curios Museum have own to Egypt to examine the great pyramid of Key-Ops. Using state-of-the-art technology they are able to determine that the lower floor of the pyramid is constructed from a series of straight line walls, which intersect to form numerous enclosed chambers. Currently, no doors exist to allow access to any chamber. This state-of-the-art technology has also pinpointed the location of the treasure room. What these dedicated (and greedy) archaeologists want to do is blast doors through the walls to get to the treasure room. However to minimize the damage to the artwork in the intervening chambers (and stay under their government grant for dynamite) they want to blast through the minimum number of doors. For structural integrity purposes, doors should only be blasted at the midpoint of the wall of the room being entered.

You are to write a program which determines this minimum number of doors. An example is shown below:



Input

The input will consist of one case. The 1st line will be an integer n (0 <= n <= 30) specifying number of interior walls, followed by n lines containing integer endpoints of each wall x1 y1 x2 y2. The 4 enclosing walls of the pyramid have fixed endpoints at (0, 0), (0, 100), (100, 0), (100, 100) and are not included in the list of walls. The interior walls always span from one exterior wall to another exterior wall and are arranged such that no more than two walls intersect at any point. You may assume that no two given walls coincide. After the listing of the interior walls there will be one final line containing the floating point coordinates of the treasure in the treasure room (guaranteed not to lie on a wall).

Output

Print a single line listing the minimum number of doors which need to be created, in the format shown below.

Sample Input

```
7
20 0 37 100
40 0 76 100
85 0 0 75
100 90 0 90
0 71 100 61
0 14 100 38
100 47 47 100
54.5 55.4
```

Sample Output

```
Number of doors = 2
```

Digital Lab

Assume that you work for the Digital Processing Lab. They ask you to write a program with an input binary matrix A, which contains the pattern to search on other binary matrix B. The input file include the size and elements for both A and B. The recognition process consists in scanning row by row (horizontal scanning) the matrix B, when a pattern is located on B you must mark this pattern. To mark a located pattern change 1 to 2 and 0 to * on B. The output file of your program will be the matrix B with the located patterns marked.

Input
The first line of the input contains the size of A, next lines contains the matrix A row by row, next line contains the size of B and next lines contains the matrix B row by row.

Output
The output is the matrix B with the located patterns marked.

INPUT FILE
2 2
1 0
1 1
5 5
1 1 0 0 0
0 1 1 0 0
1 0 0 1 0
1 1 1 1 0
0 0 1 1 1

Note: The input file contains the size of the matrix A, the matrix A, the size of the matrix B and the matrix B.

OUTPUT FILE
1 2 * 0 0
0 2 2 0 0
2 * 0 1 0
2 2 1 2 *
0 0 1 2 2

INPUT FILE
1 1
1
5 5
1 1 0 0 0
0 1 1 0 0
1 0 0 1 0
1 1 1 1 0
0 0 1 1 1

OUTPUT FILE
2 2 0 0 0
0 2 2 0 0

2 0 0 2 0
2 2 2 2 0
0 0 2 2 2

INPUT FILE
1 1
0
5 5
1 1 0 0 0
0 1 1 0 0
1 0 0 1 0
1 1 1 1 0
0 0 1 1 1

OUTPUT FILE
1 1 * * *
* 1 1 * *
1 * * 1 *
1 1 1 1 *
* * 1 1 1

INPUT FILE
2 6
1 0 0 1 0 1
1 1 1 0 1 0
5 5
1 1 0 0 0
0 1 1 0 0
1 0 0 1 0
1 1 1 1 0
0 0 1 1 1

OUTPUT FILE
1 1 0 0 0
0 1 1 0 0
1 0 0 1 0
1 1 1 1 0
0 0 1 1 1