

Mini Exam: Multidimensional Arrays

You can check your solutions here: <https://judge.softuni.bg/Contests/3174/Additional-Exercises>.

1. Difference of Diagonals

Print the **difference between the sums of diagonals of square matrix** (absolute value).

| | 0 | 1 | 2 |
|---|----|---|-----|
| 0 | 11 | 2 | 4 |
| 1 | 4 | 5 | 6 |
| 2 | 10 | 8 | -12 |

primary diagonal
sum = $11 + 5 - 12 = 4$

| | 0 | 1 | 2 |
|---|----|---|-----|
| 0 | 11 | 2 | 4 |
| 1 | 4 | 5 | 6 |
| 2 | 10 | 8 | -12 |

secondary diagonal
sum = $4 + 5 + 10 = 19$

Input

- At the **first line**, you will be given an integer **N** – which is the size of the square matrix
- At the next **N lines**, you will be given the values for **every row** – **N** numbers separated by a space

Output

- Print the **absolute** difference between the **sums** of the first and the second diagonal

Examples

| Input | Output | Comments |
|----------------------------------|--------|---|
| 3 11 1 4 8 2 9 10 2 -12 | 15 | First diagonal: sum = $11 + 2 + (-12) = 1$ Second diagonal: sum = $4 + 2 + 10 = 16$ The absolute difference: $ 1 - 16 = 15$ |

2. Matrix Changings

Write a program that reads a string matrix from the console and makes certain changes of its elements. First you will read the **dimensions** of the matrix and then the **data** of it. You will receive commands with action to perform and coordinates in the matrix like that: "**exchange row1 col1 row2 col2**". The valid command should start with the "**exchange**" and continues with four valid **coordinates** (no less or more than 4). **After every made exchange** of values at the given coordinates (cell [row1, col1] with cell [row2, col2]), you have to **print the matrix**. This is how you'll be able to check if the operation was performed correctly.

If you receive **invalid command** (doesn't contain the keyword "exchange", has fewer or more coordinates entered or the given coordinates do not exist) print "**The input is invalid!**" and move on to the next command.

Your program should finish when you receive command "**END**".

Examples

| Input | Output |
|-------|--------|
|-------|--------|

| | |
|---|---|
| 2 3 1 2 3 4 5 6 exchange 0 0 1 1 exchange 10 9 8 7 exchange 0 1 1 0 END | 5 2 3 4 1 6 The input is invalid! 5 4 3 2 1 6 |
| 1 2 Two Worlds 0 0 0 1 exchange 0 0 0 1 exchange 0 1 0 0 END | The input is invalid! Worlds Two Two Worlds |

3. * Digger

We get as input **the size** of the **square field** in which our digger moves. After that we will receive the commands which represent the directions in which the digger should move. The digger **starts** from position – ‘s’. The commands will be: **down**, **up**, **right** and **left**. If the digger has reached a side edge of the field and the next command indicates that he has to get out of the field, he must **remain on his current position and ignore the current command**. The possible characters that may appear on the screen are:

- * – a regular position on the field.
- e –end of the route.
- r - rock
- s - the **start** place where the **digger begins**

If the digger finds a rock, he collects it and **replaces it with '*'**. Track the **count of the collected rocks**. If the digger collects all of the rocks in the field, the program stops and you have to print: **"All rocks are collected!! ({rowIndex}, {colIndex})"**.

If the digger **gets on 'e' the game is over (the program stops)** and you have to print: **"Game over! ({rowIndex}, {colIndex})"**.

If there are no more commands and none of the above cases had happened, you have to print the following message: **"{remainingRocks } rocks left. ({rowIndex}, {colIndex})"**.

Input

- **Field size** is an integer number.
- **Commands to move** the digger are an array of strings separated by " ".
- **The field: only the following characters (*, e, r, s)**, separated by (" ");

Output

- Types of output:
 - If all the rocks have been collected, print: **"All rocks are collected!! ({rowIndex}, {colIndex})"**
 - If the end is reached, stop the program and print: **"Game over! ({rowIndex}, {colIndex})"**
 - If none of the above is true and there are no more commands, print: **"{totalRocks} rocks left. ({rowIndex}, {colIndex})"**

Constraints

- You will always have only one ‘s’ in the matrix.
- Allowed working time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

| Input | Output |
|---|---------------------------------|
| 3 right * r r s e * * * * | Game over! (1, 1) |
| 3 down down right right s * * * * * r r r | All rocks are collected! (2, 2) |
| 6 left left down right up left left down down down * * * * * e * * * r * * * r s * * * * * * * r * * * r * * * r * * * | 3 rocks left. (5, 0) |