Homework: Multidimensional Arrays, Sets, Dictionaries

This document defines the homework assignments from the "Advanced C#" Course @ Software University. Please submit as homework a single **`zip`** / **`rar`** / **`7z`** archive holding the solutions (source code) of all below described problems.

Problem 1. Fill the Matrix

Write two programs that **fill** and print a **matrix** of size **N x N**. Filling a matrix in the regular pattern (**top to bottom** and **left to right**) is boring. Fill the matrix as described in both patterns below:

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

Problem 2. Maximal Sum

Write a program that reads a rectangular integer matrix of size **N x M** and finds in it the square **3 x 3** that **has maximal sum of its elements**.

On the first line, you will receive the rows **N** and columns **M**. On the next **N lines** you will receive **each row with its columns**.

Print the **elements** of the 3 x 3 square as a matrix, along with their **sum**.

Input	Matrix	Output
4 5 1 5 5 2 4 2 1 4 14 3 3 7 11 2 8 4 8 12 16 4	1 5 5 2 4 2 **1 4 14** 3 3 **7 11 2** 8 4 **8 12 16** 4	Sum = 75 1 4 14 7 11 2 8 12 16

Problem 3. Matrix shuffling

Write a program which reads a string matrix from the console and performs certain operations with its elements. User input is provided like in the problem above – first you read the dimensions and then the data. Remember, you are not required to do this step first, you may add this functionality later.

Your program should then receive commands in format: "swap x1 y1 x2 y2" where x1, x2, y1, y2 are coordinates in the matrix. In order for a command to be valid, it should start with the "swap" keyword along with four valid coordinates (no more, no less). You should swap the values at the given coordinates (cell [x1, y1] with cell [x2, y2]) and print the matrix at each step (thus you'll be able to check if the operation was performed correctly).

If the command is not valid (doesn't contain the keyword "swap", has fewer or more coordinates entered or the given coordinates do not exist), print "Invalid input!" and move on to the next command. Your program should finish when the string "END" is entered. Examples:

Input	Output
2 3 1 2 3 4 5 6 swap 0 0 1 1 swap 10 9 8 7 swap 0 1 1 0 END	*(after swapping 1 and 5):* 5 2 3 4 1 6 Invalid input! *(after swapping 2 and 4):* 5 4 3 2 1 6
1 2 Hello World 0 0 0 1 swap 0 0 0 1 swap 0 1 0 0 END	Invalid input World Hello Hello World

Problem 4. Sequence in Matrix

We are given a matrix of strings of size N x M. Sequences in the matrix we define as sets of several neighbour elements located on the same **line, column or diagonal**. Write a program that finds the longest sequence of equal strings in the matrix. Examples:

Matrix	Output
ha fifi ho hi fo **ha** hi xx xxx ho **ha** xx	ha, ha, ha

Matrix	Output
s qq **s** pp pp **s** pp qq **s**	s, s, s

Problem 5. Collect the Coins

Working with multidimensional arrays can be (and should be) fun. Let's make a game out of it.

You receive the layout of a **board** from the console. Assume it will always have **4 rows** which you'll get as strings, each on a separate line. Each character in the strings will represent a **cell** on the board. Note that the strings may be of different length.

You are the player and start at the top-left corner (that would be position **[0, 0]** on the board). On the fifth line of input you'll receive a string with movement commands which tell you where to go next, it will contain only these four characters – '**>**' (move right), '**<**' (move left), '**^**' (move up) and '**v**' (move down).

You need to keep track of two types of events – collecting coins (represented by the symbol '**$**', of course) and hitting the walls of the board (when the player tries to move off the board to invalid coordinates). When all moves are over, **print the amount of money** collected and the **number of walls hit**. Example:

Input	Output	Comments
Sj0u$hbc $87yihc87 Ewg3444 $4$$ V>>^^>>>VVV< <<	Coins collected: 2 Walls hit: 2	Starting from (0, 0), move down (coin), twice right, up, up again (wall), three times right (coin on second move), twice down, down again (wall), twice to the left – game over (no more moves). Total of two coins collected and two walls hit in the process.

Problem 6. Count Symbols

Write a program that reads some text from the console and counts the occurrences of each character in it. Print the results in **alphabetical** (lexicographical) order. Examples:

Input	Output
SoftUni rocks	: 1 time/s S: 1 time/s U: 1 time/s c: 1 time/s f: 1 time/s i: 1 time/s k: 1 time/s n: 1 time/s o: 2 time/s r: 1 time/s s: 1 time/s t: 1 time/s
Did you know Math.Round rounds to the nearest even integer?	: 9 time/s .: 1 time/s ?: 1 time/s D: 1 time/s M: 1 time/s R: 1 time/s a: 2 time/s d: 3 time/s e: 7 time/s g: 1 time/s h: 2 time/s i: 2 time/s k: 1 time/s n: 6 time/s o: 5 time/s r: 3 time/s s: 2 time/s t: 5 time/s u: 3 time/s v: 1 time/s w: 1 time/s y: 1 time/s
Uvh34yt78y78y7Y&T^^DFt362t62thfwuihhYG&GY2	&: 2 time/s 2: 3 time/s 3: 2 time/s 4: 1 time/s

	6: 2 time/s 7: 3 time/s 8: 2 time/s D: 1 time/s F: 1 time/s G: 2 time/s T: 1 time/s U: 1 time/s Y: 3 time/s ^: 2 time/s f: 1 time/s h: 4 time/s i: 1 time/s t: 4 time/s u: 1 time/s v: 1 time/s w: 1 time/s y: 3 time/s

Problem 7. Phonebook

Write a program that receives some info from the console about **people** and their **phone numbers**.

You are free to choose the manner in which the data is entered; each **entry** should have just **one name** and **one number** (both of them strings).

After filling this simple phonebook, upon receiving the **command** "**`search`**", your program should be able to perform a search of a contact by name and print her details in format "**`{name} -> {number}`**". In case the contact isn't found, print "**`Contact {name} does not exist.`**" Examples:

Input	Output
Nakov-0888080808 **search** Mariika Nakov	Contact Mariika does not exist. Nakov -> 0888080808
Nakov-+359888001122 RoYaL(Ivan)-666 Gero-5559393 Simo-02/987665544 **search** Simo simo RoYaL RoYaL(Ivan)	Simo -> 02/987665544 Contact simo does not exist. Contact RoYaL does not exist. RoYaL(Ivan) -> 666

* **Bonus:** What happens if the user enters the same name twice in the phonebook? Modify your program to keep **multiple phone** numbers per contact.

Problem 8. Night Life

Being a nerd means writing programs about night clubs instead of actually going to ones. Spiro is a nerd and he decided to summarize some info about the most popular night clubs around the world.

He's come up with the following structure – he'll summarize the data by **city**, where each city will have a **list of venues** and each **venue** will have a **list of performers**. Help him finish the program, so he can stop staring at the computer screen and go get himself a cold beer.

You'll receive the input from the console. There will be an **arbitrary** number of lines until you receive the string "**END**". Each line will contain data in format: "**city;venue;performer**". If either **city**, **venue** or **performer don't exist** yet in the database, **add them**. If either the city and/or venue **already exist**, **update their info**.

Cities should remain in the **order in which they were added**, **venues** should be **sorted alphabetically** and **performers** should be **unique** (per venue) and also **sorted alphabetically**.

Print the data by **listing the cities** and for each city its **venues** (on a new line starting with "**->**") and **performers** (separated by comma and space). Check the examples to get the idea. And grab a beer when you're done, you deserve it. Spiro is buying.z

Input	Output
Sofia;Biad;Preslava Pernik;Stadion na mira;Vinkel New York;Statue of Liberty;Krisko Oslo;everywhere;Behemoth Pernik;Letishteto;RoYaL Pernik;Letishteto;RoYaL Pernik;Stadion na mira;Bankin Pernik;Stadion na mira;Vinkel END	Sofia ->Biad: Preslava Pernik ->Letishteto: RoYaL ->Stadion na mira: Bankin, Vinkel NewYork ->Statue of Liberty: Krisko Oslo ->everywhere: Behemoth

Problem 9. * Terrorists Win!

This problem is from the Java Basics Exam (7 January 2015). You may check your solution here.

On de_dust2 terrorists have planted a bomb (or possibly several of them)! Write a program that sets those bombs off!

A bomb is a string in the format **|...|**. When set off, the bomb destroys all characters inside. The bomb should also destroy **n** characters to the left and right of the bomb. **n** is determined by the **bomb power** (the **last digit of the ASCII sum** of the characters inside the bomb). Destroyed characters should be replaced by '**.**' (dot). For example, we are given the following text:

```
prepare|yo|dong
```

The bomb is **|yo|**. We get the bomb power by calculating the last digit of the sum: **y** (121) + **o** (111) = 23**2**. The bomb explodes and destroys itself and **2** characters to the left and **2** characters to the right. The result is:

```
prepa........ng
```

Input

The input data should be read from the console. On the first and only input line you will receive the text.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The destroyed text should be printed on the console.

Constraints

- The length of the text will be in the range [1...1000].
- The bombs will hold a number of characters in the range [0...100].
- Bombs will not be nested (i.e. bomb inside another bomb).
- Bomb explosions will never overlap with other bombs.
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
`prepare|yo|dong`	`prepa........ng`

Input	Ouput
`de_dust2 |A| the best |BB|map!`	`de_d.............bes........p!`

Problem 10. * Plus-Remove

This problem is originally from the JavaScript Basics Exam (24 November 2014). You may check your solution here.

You are given a sequence of **text lines**, holding symbols, small and capital Latin letters. Your task is to **remove all "plus shapes"** in the text. They may consist of small and capital letters at the same time, or of any symbol. All of the **X shapes** below are **valid**:

a aaa a	B BBB B	T TtT T	p PPp P	& &&& &	* *** *	etc.

Every **"plus shape"** is 3 by 3 symbols crossing each other on 3 lines. Single **"plus shape"** can be part of **multiple "plus shapes"**. If new **"plus shapes"** are formed after the first removal **you don't have** to remove them.

Input

The input data will be received from the console. It consists of variable number of lines. The input ends with the string "**END**" (it should not be taken into account in the program logic).

Output

Print at the console the input data after removing all **"plus shapes"**.

Constraints

- Each input line will hold between 1 and 100 Latin letters.
- The number of input lines will be in the range [1 ... 100].
- Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

Examples

Input	Output
ab**l5 bBb*555 absh*5 ttHHH ttth END	a**l * as* tt ttt

Input	Output
888**t* 8888ttt 888ttt< *8*0t>>hi END	8*** << **0>>hi

Input	Output
@s@a@p@una p@@@@@@@@dna @6@t@*@*ego vdig*****ne6 li??^*^* END	@sapuna p@dna @6tego vdigne6 li??^^

Problem 11. * String Matrix Rotation

This problem is originally from the JavaScript Basics Exam (28 July 2014). You may check your solution here.

You are given a **sequence of text lines**. Assume these text lines form a **matrix of characters** (pad the missing positions with spaces to build a rectangular matrix). Write a program to **rotate the matrix** by 90, 180, 270, 360, ... degrees. Print the result at the console as sequence of strings. Examples:

Input	Rotate(90)	Rotate(180)	Rotate(270)
hello softuni exam h e l l o s o f t u n i e x a m	e s h x o e a f l m t l u o n i	m a x e i n u t f o s o l l e h	i n o u l t m l f a e o x h s e

Input

The input is read from the console:

- The first line holds a command in format "**Rotate(X)**" where **X** are the degrees of the requested rotation.
- The next lines contain the **lines of the matrix** for rotation.
- The input ends with the command "END".

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print at the console the **rotated matrix** as a sequence of text lines.

Constraints

- The rotation **degrees** is positive integer in the range [0...90000], where **degrees** is **multiple of 90**.
- The number of matrix lines is in the range [1...**1 000**].
- The matrix lines are **strings** of length 1 ... 1 000.
- Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

Examples

Input	Output

Input	Output

Input	Output

Rotate(90) hello softuni exam END	esh xoe afl mtl uo n i

Rotate(180) hello softuni exam END	maxe inutfos olleh

Rotate(270) hello softuni exam END	i n ou ltm lfa eox hse

Input	Output
Rotate(720) js exam END	js exam

Input	Output
Rotate(810) js exam END	ej xs a m

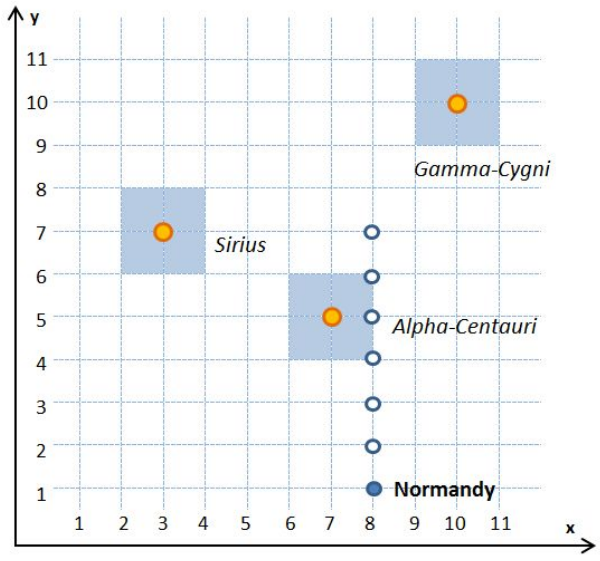
Input	Output
Rotate(0) js exam END	js exam

Problem 12. * To the Stars!

This problem is from the JavaScript Basics Exam (4 September 2014). You may check your solution here.

The year is 2185 and the SSR Normandy spaceship explores our galaxy. Unfortunately, the ship suffered severe damage in the last battle with Batarian pirates, and her navigation system is broken. Your task is to write a JavaScript program to help the Normandy safely navigate through the stars back home.

The navigation field is a 2D grid. You are given the names of **3 star systems**, along with **their coordinates(s_x, s_y)** and **the Normandy's initial coordinates(n_x, n_y)**. Assume that a **star's coordinates are in the center of a 2x2 rectangle**. The Normandy **always moves in an upwards direction, 1 unit every turn**. Your task is to inform the Normandy of its current location during its movement.



The Normandy can **only be at one location** at a time. The possible locations are "**<star1 name>**", "**<star2 name>**", "**<star3 name>**" and "**space**". In other words, if the Normandy is in the range of Alpha-Centauri, her location is "alpha-centauri". If she's not in the range of any star system, her location is just "space".

Star systems will **NOT** overlap.

Example: the Normandy's initial location is at (8, 1). There, she in outside of any star system, so she is in "space". She starts moving up and her next two locations at (8, 2) and (8, 3) are again in "space". After that, at (8, 4), (8, 5), (8, 6) she is in the range of Alpha-Centauri – therefore, she is in "alpha-centauri". Her final location (8, 7) is outside any star, and her location is "space".

Input

The input is passed to the first JavaScript function found in your code as **array of several arguments**:

- The first arguments will contain each star system's name and coordinates in the format "**<name>** **<x>** **<y>**", separated by spaces. The **name will be a single word, without spaces**.

- The fourth argument will contain the Normandy's initial coordinates in the format "<**x**> <**y**>", separated by spaces.
- The fifth, last argument, will contain the number **n** – the number of turns the Normandy will be moving.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output consists of **n + 1** lines – the Normandy's **initial** location, plus the **locations she was in during her movement**, each on a separate line. All locations must be printed **lowercase**.

Constraints

- The grid dimensions will be no larger than 30x30.
- All star systems will be squares with a fixed size: 2x2.
- The turns will be no more than 20.
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
Sirius 3 7 Alpha-Centauri 7 5 Gamma-Cygni 10 10 8 1 6	space space space alpha-centauri alpha-centauri alpha-centauri space

Input	Output
Terra-Nova 16 2 Perseus 2.6 4.8 Virgo 1.6 7 2 5 4	perseus virgo virgo virgo space

Problem 13. * Activity Tracker

This problem is from the Java Basics Exam (3 September 2014). You may check your solution here.

You are part of the server side application team of brand new and promising activity tracking company. Their product "The Spy" is constantly sending data to the server. The data represents the distance walked in meters for every user in format:

- 24/07/2014 Angel 4600
- 24/07/2014 Pesho 3200
- 25/07/2014 Angel 6500
- 01/08/2014 Pesho 5600
- 03/08/2014 Ivan 11400

Write a program to aggregate the data and print the **activity of each user per month**. The months should be represented in ascending order. The users should be ordered alphabetically and the data should be represented in the following way: **<month>: <user>(<distance>), <user>(<distance>),...** For the data above the output should be:

- 7: Angel(11100), Pesho(3200)
- 8: Ivan(11400), Pesho(5600)

Input

The input comes from the console. At the first line a number **n** stays which says how many data lines will follow. Each of the next n lines holds a data in format **<date> <user> <distance>**. The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print **one line for each month** (months are in ascending order). For each month print the users and the sum of distances for each one of them, in descending order in format **<month>: <user>(<distance>), <user>(<distance>),...**

Constraints

- The **count** of the data lines **n** is in the range [1...1000].
- The **<date>** is a standard date in format **dd/mm/yyyy** where **dd** is the day of the month, **mm** is the numeric representation of the month and **yyyy** is the full numeric represenation of the year.
- The **<user>** consists of only of **Latin characters**, with length of [1...20].
- The **<distance>** is floating point number in the range [1...1000].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Example

Input	Output	Input	Output
5 24/07/2014 Angel 4600 24/07/2014 Pesho 3200 25/07/2014 Angel 6500 01/08/2014 Pesho 5600 03/08/2014 Ivan 11400	7: Angel(11100), Pesho(3200) 8: Ivan(11400), Pesho(5600)	3 10/07/2014 Nakov 10345 15/07/2014 Nakov 8765 20/07/2014 Nakov 4532	7: Nakov(23642)