Exercise: Recursion and Combinatorial Problems

This document defines the lab for "Algorithms – Fundamentals (C#)" course @ Software University.

Please submit your solutions (source code) of all below-described problems in Judge.

1. Reverse Array

Write a program that reverses and prints an array. Use **recursion**.

Examples

Input	Output
1 2 3 4 5 6	6 5 4 3 2 1
-1 0 1	1 0 -1

2. Nested Loops To Recursion

Write a program that simulates the execution of n nested loops **from 1 to n** which prints the values of all its iteration variables at any given time on a single line. **Use recursion.**

Examples

Input	Output
2	1 1 1 2 2 1 2 2
3	1 1 1 1 1 2 1 1 3 1 2 1 1 2 2 … 3 2 3 3 3 1 3 3 2 3 3 3

3. Connected Areas in a Matrix

Let's define a **connected area** in a matrix as an area of cells in which there is a **path between every two cells**.

Write a program to find **all** connected areas in a matrix.

Input

- On the first line, you will get the **number of rows**.
- On the second line, you will get the **number of columns**.
- The rest of the input will be the **actual matrix**.

Output

- Print on the console the **total number of areas found**.
- On a separate line for each area print its **starting coordinates** and **size**.
- **Order** the areas by size (in descending order) so that the **largest area is printed first**.
    - If several areas have the same size, order them **by their position**, first by the row, then by the column of the top-left corner.
    - If there are two connected areas of the same size, the one which is above and/or to the left of the other will be printed first.

Examples

Input	Output
4 9 ---*---*- ---*---*- ---*---*- ----*-*--	Total areas found: 3 Area #1 at (0, 0), size: 13 Area #2 at (0, 4), size: 10 Area #3 at (0, 8), size: 5
5 10 *--*---*-- *--*---*-- *--*****-- *--*---*-- *--*---*--	Total areas found: 4 Area #1 at (0, 1), size: 10 Area #2 at (0, 8), size: 10 Area #3 at (0, 4), size: 6 Area #4 at (3, 4), size: 6

Hints

- Create a method to find the first traversable cell which hasn't been visited. This would be the top-left corner of a connected area. If there is no such cell, this means all areas have been found.
- You can create a class to hold info about a connected area (its position and size). Additionally, you can implement Comparable and store all areas found in a TreeSet.

4. Cinema

Write a program that prints all of the possible **distributions** of a group of friends in a **cinema hall**. On the **first line,** you will be given all of the friends' **names** separated by **comma and space**. Until you receive the command **"generate"** you will be given some of those friends' **names** and a **number of the place** that they want to have. (format: **"{name} - {place}"**) So here comes the tricky part. Those friends **want** only to **sit** in the place that they **have chosen**. They **cannot sit in other places**. For more clarification see the examples below.

Output

Print all the **possible ways to distribute the friends** having in mind that some of them want a particular place and they will sit there only. The **order** of the output does **not matter**.

Constrains

- The friends' **names** and the **number** of the place will always be valid.

Examples

Input	Output	Comments
Peter, Amy, George, Rick Amy - 1 Rick - 4 generate	Amy Peter George Rick Amy George Peter Rick	Amy only wants to sit on the first seat and Rick wants to sit on the fourth, so we only switch the other friends
Garry, Liam, Teddy, Anna, Buddy, Simon Buddy - 3 Liam - 5 Simon - 1 generate	Simon Garry Buddy Teddy Liam Anna Simon Garry Buddy Anna Liam Teddy Simon Teddy Buddy Garry Liam Anna Simon Teddy Buddy Anna Liam Garry Simon Anna Buddy Garry Liam Teddy Simon Anna Buddy Teddy Liam Garry	

5. School Teams

Write a program that receives the **names** of **girls** and **boys** in a class and generates **all possible ways** to create **teams** with **3 girls** and **2 boys**. Print each team on a **separate line** separated by comma and space **", "** (**first** the **girls then** the **boys**). For more clarification see the examples below

Note: **"Linda, Amy, Katty, John, Bill"** is the **same as "Linda, Amy, Katty, Bill, John"**; so print only **the first case**

Input

- On the **first line,** you will receive the **girls'** names separated by comma and space **", "**.
- On the **second line,** you will receive the **boys'** names separated by comma and space **", "**.

Output

- On **separate lines** print all the possible **teams** with exactly **3 girls** and **2 boys** separated by comma and space and starting with the girls.

Constrains

- There will always be **at least 3 girls** and **2 boys** in the input.

Examples

Input	Output
Linda, Amy, Katty John, Bill	Linda, Amy, Katty, John, Bill
Lisa, Yoana, Marta, Rachel George, Garry, Bob	Lisa, Yoana, Marta, George, Garry Lisa, Yoana, Marta, George, Bob Lisa, Yoana, Marta, Garry, Bob Lisa, Yoana, Rachel, George, Garry Lisa, Yoana, Rachel, George, Bob Lisa, Yoana, Rachel, Garry, Bob Lisa, Marta, Rachel, George, Garry Lisa, Marta, Rachel, George, Bob Lisa, Marta, Rachel, Garry, Bob Yoana, Marta, Rachel, George, Garry Yoana, Marta, Rachel, George, Bob Yoana, Marta, Rachel, Garry, Bob

6. Word Cruncher

Write a program that receives some **strings** and **forms another** string that is required. On the **first line,** you will be given **all of the strings** separated by **comma and space**. On the next line, you will be given the **string** that needs to be **formed from the given strings**. For more clarification see the examples below.

Input

- On the first line, you will receive the **strings** (separated by comma and space **", "**).
- On the next line, you will receive the **target string.**

Output

- Print each of them found ways to form the required string as shown in the examples.

Constrains

- There might be **repeating elements** in the input.

Examples

Input	Output

text, me, so, do, m, ran somerandomtext	so me ran do m text
Word, cruncher, cr, h, unch, c, r, un, ch, er Wordcruncher	Word c r un ch er Word c r unch er Word cr un c h er Word cr un ch er Word cr unch er Word cruncher
tu, stu, p, i, d, pi, pid, s, pi stupid	s tu p i d s tu pi d s tu pid stu p i d stu pi d stu pid