Contest Problems
Philadelphia Classic, Fall 2017
Hosted by the Dining Philosophers
University of Pennsylvania

dining philosophers

UNIVERSITY OF PENNSYLVANIA | COMPUTER SCIENCE CLUB

Rules and Information

This document includes 12 problems. Novice teams do problems 1-8; standard teams do problems 5-12.

Any team which submits a correct solution for any of problems 1-4 will be assumed to be a novice team. **If you are not a novice team, please skip problems 1-4.**

Problems 1-4 are easier than problems 5-8, which are easier than problems 9-12. These problems are correspondingly labeled “Novice”, “Intermediate”, and “Advanced.” Order does not otherwise relate to difficulty.

You may use the Internet only for submitting your solutions, reading Javadocs or Python documentation, and referring to any documents we link you to. You **may not** use the Internet for things like StackOverflow, Google, or outside communication.

As you may know, you can choose to solve any given problem in either **Java or Python**. We have provided stub files for all questions in both languages that takes care of the input parsing for you. **Do not modify any of the parsing or output code**. Just fill out the stub methods in each file and submit it with exactly the same name.
Do not modify any of the given methods or method headers in our stub files! They are all specified in the desired format. You may add class fields, helper methods, etc as you like, but modifying given parts will cause your code to fail in our testers.

There is no penalty for incorrect submissions. You will receive 1 point per problem solved. A team’s number of incorrect submissions will be used only as a tiebreaker.

Some problems use Java’s “long” type; if you are unfamiliar with them, they’re like an “int”, but with a (much) bigger upper bound, and you have to add “L” to the end of an explicit value assignment:

```
        long myLong = 1000000000000L;
```

Otherwise, the “long” type functions just like the “int” type.

1. Perfect - Novice

You, a recent college graduate and the head of a growing urban architecture firm, have just received your first job! Your client, the University of Pennsylvania, wants you to design the newest and most expensive dormitory on Penn's campus with a specific property (to make it the most expensive dormitory on campus): the height of the dormitory is equal to the sum of its proper divisors. That is, the height is equal to the sum of its positive divisors, not including itself. Your client hands you a list of possible building heights that they are currently examining and wants you to tell them which buildings satisfy this special property. Can you write a function that determines whether each height satisfies this property?

Input Format
The first line is the number of integers in the input file. Each following line consists of an integer 0 <= n <= 100000000.

Output Format
true if the number satisfies the given properties, false otherwise

Sample Input	**Sample Output**	**Explanation**
3	false	The proper divisor of 3 is 1. 1 != 3, so 3 does not satisfy the given properties.
6	true	The proper divisors of 6 are 1, 2, 3. 1 + 2 + 3 = 6, so 6 satisfies the given properties.
4	false	The proper divisors of 4 are 1, 2. 1 + 2 = 3 != 4, so 4 does not satisfy the given properties.

2. Subway System - Novice

As the head of the city's public transportation system, citizens of the city depends on you to get around. This isn't just a matter of convenience; a reliable transit system is critical to the urban economy. Delayed subways and busses mean missed meetings and wasted time! For example, in New York City, subway delays cost the city **$389 million** annually![1]

In an effort to improve the city's transit system, you want to evaluate the on-time performance of each route. You compute on-time performance with the following formula:

On-time performance = 100 * Vehicles that Arrive on Time / Total Services

You are also given a threshold that a route's performance must pass in order for it to be deemed "reliable". Can you compute on-time performance and check whether it passes the threshold? You do not need to worry about non-integer on-time performance values.

Input format
The first line denotes the number of vehicles that arrive on time. The second line denotes the number of total services. The third line denotes the threshold.

Output format
Output a boolean where it is true if on-time performance is above or equal to the threshold and false otherwise.

Sample Input	**Sample Output**	**Explanation**
10 20 65	false	To compute on-time performance: (10 / 20) * 100 = 50 Since 50 < 65, on-time performance is below the threshold and we output False.
5 50 10	true	To compute on-time performance: (5 / 50) * 100 = 10 Since 10 = 10, on-time performance matches the threshold and we output True.

[1] https://ny.curbed.com/2017/10/2/16396624/mta-subway-delay-nyc-economy-loss

3. Prank Planning - Novice

It is April Fool's Day and you would like to come up with a prank to fool the city. You invite your prank planning committee to brainstorm before the event, and they come up with the following prank.

The idea is to choose some **N** street signs in the city, and reverse the names on those street signs. You must now actually implement this prank.

Can you implement the April Fool's prank?

Input Format
You will be given an integer **N** representing the number of street signs you wish to change. The next **N** lines denote each street's name.

Output Format
You should output **N** lines where each line contains the street names in reverse in the same order as they were in the input.

Sample Input	**Sample Output**	**Explanation**
2 walnut chestnut	tunlaw tuntsehc	walnut in reverse is tunlaw chestnut in reverse is tuntsehc
1 oak	kao	oak in reverse is kao.

4. Parking Spots - Novice

The city has been divided into several (possibly overlapping) neighborhoods. Each neighborhood is made up of a starting street and an ending street. You and your planning committee are trying to place parking spots all over the city. For the sake of fairness, you would like to place a parking spot in each neighborhood, so that everyone living in a given neighborhood can park in the same neighborhood.

However, the resources to build parking spots at your disposal are limited. You would like to make the most efficient use of resources, so you do not want to build more than one parking spot in the same neighborhood. Given a list of all the neighborhoods with their start and end streets, can you output a consolidated list of streets to place parking spots such that all the neighborhoods are covered by a parking spot?

Input Format

The first line is a single integer **N** denoting the number of neighborhoods.

The next **N** lines each contain two integers, the first integer denotes the starting street of the neighborhood and the second integer denotes the ending street of the neighborhood.

Sample Input	**Sample Output**	**Explanation**

4
1 3
2 6
8 10
15 18
	[[1, 6], [8, 10], [15, 18]]	The neighborhoods [1,3] and [2,6] are combined to form [1,6], so a parking spot can be built in the combined neighborhood. The other neighborhoods do not overlap and are hence left as they were.

3
1 3
3 5
5 7
	[[1, 7]]	All three neighborhoods overlap, so we just need one parking spot.

5. Tower Tycoon - Intermediate

As head architect of your architecture firm, you have been tasked with designing a new residency for residents that use to live in an older retirement home. Each resident has been assigned to a specific floor, and each floor has only one resident. You want the order of residents in the building to match the order they were in previously, so you ask each resident to tell you their old floor number. Unfortunately, while none of the residents remember their old floor number, they each remember their position in the building relative to some of their neighbors. For example, Dorothy remembers that she is somewhere above John and Grace, but somewhere below Jim. Given several lists of each of the members relative positions in order from lowest to highest, write a program that outputs the order of residents from bottom floor to top. You can assume that there is always exactly one correct ordering of the floors.

Input Format
The number of lists, followed by each list of names. The names are separated by one space.

Output Format
Output a list of participants in order from lowest floor to highest floor.

Sample Input	**Sample Output**	**Explanation**
3 sam molly jim sam tim molly jim rex	[sam, tim, molly, jim, rex]	sam < tim, and tim < molly, and molly < jim, and jim < rex
4 candace rob john john tom rob tom candace tom jacob	[candace, rob, john, tom, jacob]	candace < rob < john, and john < tom, and tom < jacob

6. The Ruby Bridge - Intermediate

You are in charge of a new, shiny bridge that will connect two cities that have been at long war with each other: Emacville and Vimopolis. The bridge will feature numerous large grand pillars and trees that will hopefully ease up the previous tension between the cities.

However, there is one slight issue: due to the terrain of these cities near the water, the bridge must be made of ruby, and unfortunately ruby is very difficult to find. Good news however: you and your team have discovered a giant strip of ruby in the sea. You wish to place pillars and trees on this bridge; however, the ruby will break if an unequal number of pillars and trees are placed on the ruby. Therefore, you want to create the longest possible bridge such that the ruby bridge will not break.

You are allowed to make up to 2 cuts to the strip of ruby, and we will only keep the strip that is between those two cuts. The ruby already has preset spots to place pillars, and preset spots to place trees. Spots are numbered from 0 to N - 1, where N is the length of the ruby strip, and all the spots are evenly separated along a straight line. Help us make a decision about how to make the longest and best bridge possible.

Input Format
The first line contains the integer N.
The next line contains a sequence of N characters, which will either be 'P' or 'T', representing where the pillars and trees will be placed, respectively.

Output Format
Output an integer containing the length of the longest appropriate cut. The cuts are inclusive, meaning that cutting at position i will include position i in the ruby strip.

Sample Input	**Sample Output**	**Explanation**
10 PTPPTTPTTP	10	The whole array is the longest possible valid bridge.
11 PTPTPTPPPTP	6	The first six characters form the biggest valid bridge.
3 PTP	2	

7. Stressful Park - Intermediate

The city has an old park that is notorious for its windy paths and maze-like structure. In addition to the park's complicated layout, certain sections of the park take longer to traverse due to hilly terrain, lakes, etc. Unsurprisingly, citizens who regularly visit the park have filed complaints about getting lost in the park and not being able to find their way to the exit in a reasonable amount of time. As a result, your boss has assigned you the task of marking the shortest paths from given locations in the park to their nearest exit.

You are given a 2D array of size N x M where index (0, 0) is the starting location and index (N - 1, M - 1) is the nearest exit. The value in each cell of the array indicates how many minutes it takes to traverse that section of the park. Note that the values at index (0, 0) and (N - 1, M - 1) will be 0. Your job is to find the shortest number of minutes it would take to reach the nearest exit. For simplicity sake, you are only allowed to move downward and rightward throughout the array. Also, you do not need to worry about outputting the shortest path itself.

Input Format
First line contains two integers, N and M.
The next N lines will contain M integers each, where the Mth integer on the Nth line will contain the value of the array at index (N - 1, M - 1).

Output Format
Output a single integer representing the shortest number of minutes to reach the exit.

Sample Input	**Sample Output**	**Explanation**

5 4
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
4 5 6 0
	21	All paths where you go down or right only will take 1 + 2 + 3 + 4 + 5 + 6 = 21 minutes.

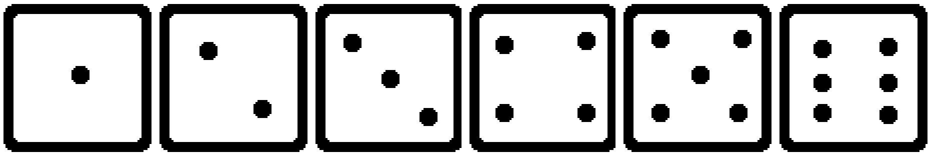
6 6
0 1 1 2 9 9
4 1 1 9 9 9
3 9 9 9 9 9
1 1 1 1 9 9
9 9 9 1 1 1
9 9 9 9 9 0
	14	The shortest path is down 3, right 3, down 1, right 2, down 1, which takes 4 + 3 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 14 minutes.

3 5
0 9 1 1 1
1 9 1 9 1
	11	Note that we cannot move upward or leftward. The shortest path is down 2, right

1 1 1 7 0		4, which takes 1 + 1 + 1 + 1 + 5 = 9 minutes.

8. Leaves Around The Tree - Intermediate

Sometimes urban planners get bored of designing cities and take a break by playing their favorite game, Leaves Around The Tree. In this game, the players are given a collection of dice and have to give their Leaves Around The Tree value.



For example, here are a few rounds with two dice.

[1, 6] -> 0
[2, 3] -> 2
[5, 4] -> 4
[5, 5] -> 8
[3, 5] -> 6
[4, 2] -> 0

The game generalizes to more than two dice

[2, 1, 6, 4, 4] -> 0
[1, 3, 1, 5, 2] -> 6
[4, 3, 6, 1, 1, 4, 4, 3, 6, 5] -> 8
[1, 3, 1, 5, 2, 4, 3, 6, 1, 1, 4, 4, 3, 6, 5, 2, 1, 6, 4, 4] -> 14

And the file dice.txt includes some more examples!

You will be given N dice. You should compute and return the Leaves Around The Tree value for the list of dice.

Hint: Consider the properties of the N-dimensional hyperplane P described by the linear equation $A^T X = 0$ where A is a vector of the dice values and X is a vector of n variables.

$$A = \begin{pmatrix} 1 \\ 6 \\ 3 \\ \vdots \\ 5 \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$$

Just kidding. Don’t overthink this.

Input Format
Each line contains a comma separated sequence of dice rolls where each dice roll is a value of 1 to 6.

Output Format
Output a the Leaves Around the Tree value for each input sequence on a separate line.

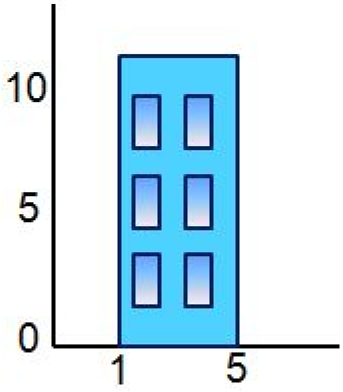
9. Skylines - Advanced

We all know that cities are defined by their skylines. While new building development is crucial to satisfy a city's growing needs, a critical concern for an urban planner is how the skyline will be affected. When investigating building projects like a new office building, high rise apartment, or stadium, an urban planner needs a way quickly check how a new building affects the skyline so they can rapidly iterate on all their ideas.

In this problem, you will write a program that will compute an approximation for the skyline of a city. You will be given a list of rectangular buildings. Each Building has three fields

- left: the x coordinate of the left wall
- right: the x coordinate of the right wall
- height: the height of the building.

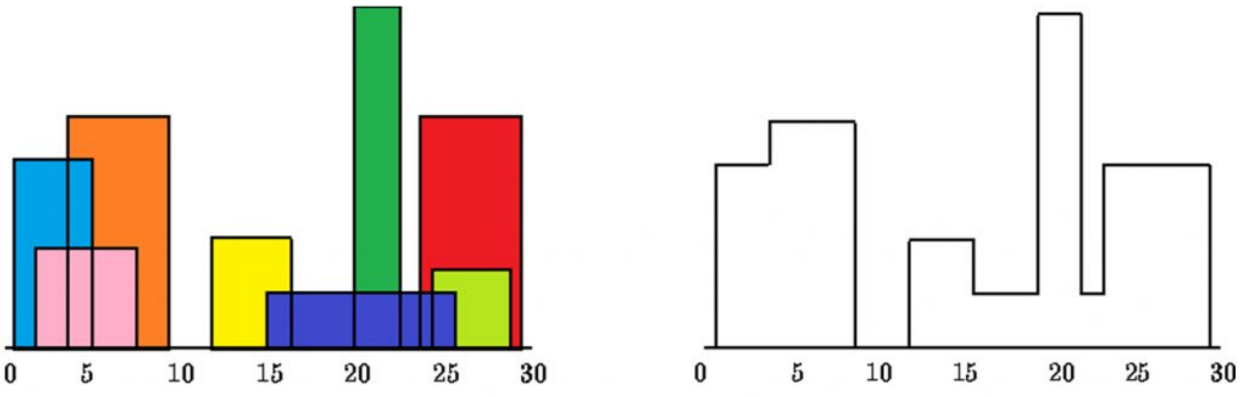
For example, the building to the right has left of 1, height of 11, and right of 5.



A skyline is a collection of rectangular strips. A Strip has two fields

- left: the x coordinate of the left side
- height: the height of the strip

For example, the collection of buildings below (left) corresponds to the skyline below (right):



Building(l, h, r) is notation for the building with left=l, height=h, and right=r.
The buildings would be represented as [Building(1,11,5), Building(2,6,7), Building(3,13,9), Building(12,7,16), Building(14,3,25), Building(19,18,22), Building(23,13,29), Building(24,4,28)]

Strip(l, h) is the notation for the strip with left=l and height=h.
The skyline would therefore be represented as [Strip(1, 11), Strip(3, 13), Strip(9, 0), Strip(12, 7), Strip(16, 3), Strip(19, 18), Strip(22, 3), Strip(23, 13), (29, 0)]

Note that every output ends in a strip of height 0. This strip represents where the skyline ends.

Input Format
The file will contain one or more test cases. Each test case starts with a number N, which is the number of buildings for that test case. Following are N lines of comma separated integers like left,height,right (each of which describes a building).

Output Format
For each test case, output the correct list of strips.

Sample Input	**Sample Output**
[Building(1,11,5), Building(2,6,7), Building(3,13,9), Building(12,7,16), Building(14,3,25), Building(19,18,22), Building(23,13,29), Building(24,4,28)]	[Strip(1, 11), Strip(3, 13), Strip(9, 0), Strip(12, 7), Strip(16, 3), Strip(19, 18), Strip(22, 3), Strip(23, 13), Strip(29, 0)]
Building(1, 11, 5)	[Strip(1, 11), Strip(5, 0)]

10. Mysterious Documents - Advanced

The city council has found a bunch of mysterious documents in a cabinet. These are relics from old founding immigrants of the city from a number of different countries. They want to be able to classify them to put them in a museum, so they have enlisted you to help them sort out which documents are written in which languages!

For this problem, your program will be given short passages (up to 5000 characters each), written in either English, Spanish, German, French, or Italian. To help you creating your program, you will be given files containing sample text in each language. For each passage, output the language that it is written in!

Input Format
Every test case consists of a single line of input containing a single passage, written in one of the five languages.

Output Format
For each passage, output either “English”, “Spanish”, “German”, “French”, or “Italian” on a single line, indicating the language of that passage.

Sample Input	Sample Output
Before Europeans arrived, the Philadelphia area was home to the Lenape (Delaware) Indians in the village of Shackamaxon. En 1681, el rey de Inglaterra Carlos II le otorgó una carta de autorización a William Penn a cambio de la anulación de una deuda que el gobierno le debía a su padre. Philadelphia war nach New York (1788–1790) die zweite Hauptstadt der Vereinigten Staaten von Amerika (1790–1800), bis zur Fertigstellung der neuen Hauptstadt Washington, D.C. Dans les années 1770, Philadelphie devint l'un des principaux foyers de la Révolution américaine. Fondata nel 1682 dal quacchero William Penn, Filadelfia è una delle più antiche città degli Stati Uniti d'America, e fra la fine del XVIII secolo e l'inizio del XIX fu la città più grande del Paese.	English Spanish German French Italian

11. Pattern Matching - Advanced

All great cities require some top-down planning to ensure a sane and robust urban design. For example, we would never want to have a night club in the middle of a residential neighborhood nor would we want to have a school in the middle of the business district. Zoning, the process of dividing land into zones (e.g. residential, industrial, mixed-use), is critical to any well-functioning city.

In this problem, you will be given the layout of a city represented as a string and a zoning code represented as a pattern. The city matches the zoning code if there is some way of partitioning the parts of the city so that a single part always conform to the same zoning code.

For example, a city described by the string “abcxyxabcxyx” would satisfy the pattern “ABAB” (where A = “abc” and B = “xyz”). In a sense, the part of the city composed of “abc” consistently satisfies the zoning code for A while the part composed of “xyz” consistently satisfies the zoning code of B.

You will given an arbitrary string containing alphabetical characters and a pattern composed of uppercase alphabetical characters. You should return if the string satisfies the pattern.

Input Format
The file will contain one or more test cases. Each corresponds to one test case. Each line contains the pattern followed by a comma and followed by the string.

Output Format
For each test case, output “True” if the string satisfies the pattern or “False” otherwise.

Sample Input	**Sample Output**
pattern: ABAB string: abcxyxabcxyx	True (A = abc, B = xyz)
pattern: A string: abc	True (A = abc)
pattern: AB string: abc	True (A = a, B = bc)
pattern: ABA string: abc	False
pattern: ABCABBC string: heyhellohiheyhellohellohi	True (A = hey, B = hello, C = hi)

12. Historic Reconstruction - Advanced

The city council is very interested in finding out about certain aspects of how the city's infrastructure has developed historically. Specifically, they are doing research about how the city's landmarks have become connected by roads over time, and want their questions answered as their research progresses.

The city consists of **N** landmarks (numbered from 1 to **N**), whereas the research consists of **Q** instructions, of two different types. The first type of instruction indicates that the next road built connects landmarks **a** and **b** in the city. The order which you receive instructions of the first type determine the time that each road is complete: the first road you receive is complete at time **t** = 1, the second at time **t** = 2, and so on. The second type of instruction is a query: at some point of time **t** in the past, how many other landmarks are reachable from a given landmark **x**?

While figuring all this out might be much easier if you had access to the entire set of instructions at the start, unfortunately, that is not how the research process works. The city council's research depends on the answers to your queries; you have to answer each query before being able to access the next query.

Input Format
For this question specifically, the queries are generated according to a formula encoded in the stub files provided. You will not need to understand the details of the formula, it is sufficient to correctly implement the three blank methods provided in the stub file, which are called as necessary.

Each test case contains four integers each: **N** (the number of landmarks), **Q** (the total number of instructions), **S** and **P** (parameters to the instruction generation process). It is guaranteed **N** and **Q** are at most 100,000.

Output Format
For each instruction of the second type, output a single integer, which represents the number of landmarks reachable from **x** at point **t**.

Note
You will need to be able to respond to each instruction much faster than linear time, otherwise your solution will probably not pass the time limit!

Sample Input	**Sample Output**	**Generated Input**	**Explanation**

5 6 4116
30
	2
	1
	4
		initialize(5 , 6)
		addRoad(3, 4, 1)
		addRoad(1, 5, 2)
		addRoad(4, 5, 3)
		reachable(5, 2)
		reachable(2, 1)
		reachable(1, 3)
			Three roads are added in total: between 3 and 4 at time 1, between 1 and 5 at time 2, and between 4 and 5 at time 3.
			
			At time 2, landmark 5 can reach 1 and itself (since the road between 4 and 5 has not been built yet), so the answer is 2.
			At time 1, landmark 2 can only reach itself.
			At time 3, landmark 1 can reach everything except landmark 2, so the answer is 4.
			
			Note that returning the wrong answer to any query will cause the research process to go wrong, and potentially generate different input for the remainder of the program!