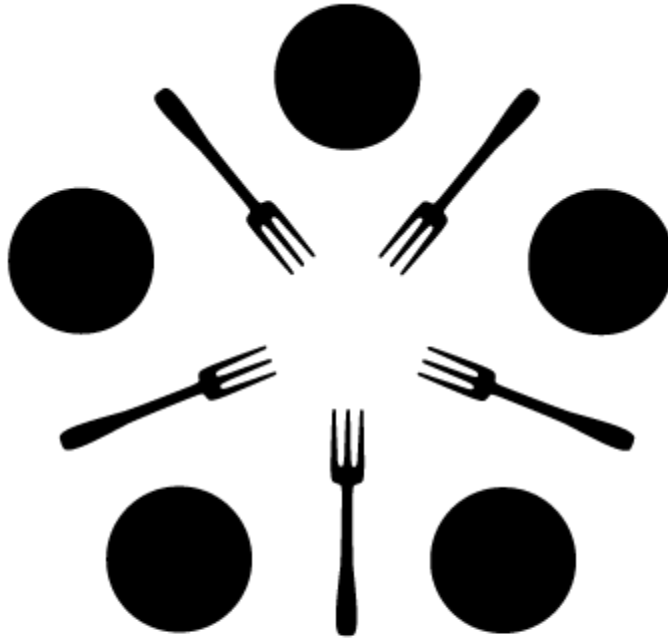


Contest Problems
Philadelphia Classic, Spring 2017
Hosted by the Dining Philosophers
University of Pennsylvania



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. Treasure Chest Scanner - Novice

In the bank's top secret vaults, gold is stored in boxes of random dimensions. During the heist, your team plans to be quick and efficient when robbing these vaults. Therefore, you have decided to construct a machine that is capable of determining the amount of gold that is stored within each of the boxes. You receive a report about how to estimate the amount of gold that is contained within each box. The report states that the amount of gold in the box is equal to $l*w*h$, where l = length, w = width, and h = height of the box. Your job is to program the machine to be able to determine how much gold is in each box using the report's complex gold estimation technique.

Input Format

You are given one line with three integers separated by spaces. The first integer represents the length of the box, the second integer represents the width of the box, and the third integer represents the height of the box.

Output

Output a single integer representing the amount of gold in the box using the formula of $\text{Gold} = l*w*h$, where l = length, w = width, and h = height.

Sample Input	Sample Output	Explanation
3 2 6	36	Gold = $3*2*6 = 36$
5 5 5	125	Gold = $5*5*5 = 125$
0 0 0	0	Gold = $0*0*0 = 0$

2. Making a Profit - Novice

In the midst of planning your heist, your team has to decide what day to steal from the National Bank of PClassic. You decide to do some research online and compile a list of previous successful heists. As you make your list, you write down the days of the week that other groups stole from the bank, and the corresponding monetary amount that they stole. Your team decides that the optimal day for your heist is the day that in the past resulted in the highest profits. As treasurer, your task is to find which day among all the previous heists resulted in the most money being stolen.

Input Format

You are given a 2D array of ints of size **7** by **n**, where **7** is the days of the week and **n** represents the total number of heists. The first two lines of the input represent the 2D array's dimensions. The following lines correspond to the contents of each row of the array.

Output Format

Output a string with the day of the week that resulted in the largest amount of money being stolen. Let row 0 = Monday, 1 = Tuesday, 2 = Wednesday, 3 = Thursday, 4 = Friday 5 = Saturday, 6 = Sunday. You can assume that all total amounts of money stolen on each day will be distinct.

Sample Input	Sample Output	Explanation
7 4 50 200 700 13 20 1000 0 0 0 0 0 0 50 0 0 10 23 15 0 0 0 0 13 0 0 13 35 68	Tuesday	The total profits for each day are: Monday: $\$50 + \$200 + \$700 + \$13 = \$963$ Tuesday: $\$20 + \$1000 = \$1020$ Wednesday: $\$0$ Thursday: $\$50 + \$10 = \$60$ Friday: $\$23 + 15 = \38 Saturday: $\$13$ Sunday: $\$13 + 35 + 68 = \116
7 2 10 20 20 30 40 70 20 20 40 60 10 10 0 0	Wednesday	The total profits for each day are: Monday: $\$10 + \$20 = \$30$ Tuesday: $\$20 + \$30 = \$50$ Wednesday: $\$40 + \$70 = \$110$ Thursday: $\$20 + \$20 = \$40$ Friday: $\$40 + \$60 = \$100$ Saturday: $\$10 + 10 = \20 Sunday: $\$0 + 0 = \0

3. Cardinal Confusion - Novice

As you are planning your heist, you realize that you need to plan out the most efficient escape route. One of your team members suggest that the best escape route is north of the bank, but the other members argue that the other three cardinal directions are the optimal routes. Hoping to avoid further dispute, you decide to look at a map and count how many safe houses are within 10 miles in the four cardinal directions. However, in the middle of looking, you decide that the best option is to find an even distribution of safe houses as to confuse any pursuer who may try to predict your escape route. You come up with a formula: $ab - cd = 1$, where each variable represents the number of safe houses in one of the four cardinal directions, to minimize the spread of safe houses and find the optimal distribution. Your task is to find how many solutions for a, b, c, d will satisfy this equation.

Input Format

You are given n , which is the upper bound of the domain, $[1, n]$, of the variables a, b, c, d , where each variable represents the number of safe houses in a distinct cardinal direction.

Output Format

Output an integer with the number of solutions to the equation $ab - cd = 1$.

Sample Input	Sample Output	Explanation
2	2	There are 2 such quadruples (2, 1, 1, 1) (1, 2, 1, 1)
3	8	There are 8 such quadruples (2, 1, 1, 1) (1, 2, 1, 1) (1, 3, 1, 2) (3, 1, 1, 2) (1, 3, 2, 1) (3, 1, 2, 1) (2, 2, 3, 1) (2, 2, 1, 3)
14	226	These were totally counted by hand yesterday.

4. Security Analysis - Novice

During your team's research into previous heists conducted on the National Bank of PClassic, you have accidentally gained access to a large set of encrypted files that may be related to the bank's security codes. Because these files are incredibly large, your team suggests to analyze common portions between files in order to efficiently determine the bank's security codes. Fortunately, your team has found the first character of each of the bank's security codes, but the files are still too large to manually determine the bank's security codes. Your intuition tells you that the longest string starting with that letter that appears in both documents is probably the code. Therefore, you are tasked with creating a computer program that when given a certain character, will take in two of these encrypted files and output the longest common substring between these two files that starts with the given character.

Input Format

You are given three lines. The first line gives the letter that longest common substring will look for. The second and third lines are the two encrypted files in string format.

Output Format

Print out the longest common substring between the two encrypted files that start with the letter.

Sample Input	Sample Output	Explanation
b babababababab pabapabapabapabap	ba	"ba" is the longest common substring that starts with b. Note that "aba" is the longest common substring, but it does not start with b.
m mommydad daddymommy	mommy	"mommy" is the longest common substring that starts with m.
y potatopiefails bankheistwillfail		There is no common substring that starts with y, so print "".

5. Unlock All Safes - Intermediate

While planning your heist on the National Bank of PClassic, you realize that you will need to unlock all their safes. Each safe is protected by a different number of locks. While you do not know the exact combination of the safe, you do know the number of locks guarding a given safe. Each safe has a grid of locks which must be opened in a clockwise spiral pattern starting from the upper left corner. Specifically, you will be given a grid of locks IDs, and you must output the ordering of the IDs such that it will unlock a given safe. Can you develop an algorithm that can help you unlock any safe?

Input Format

You are given an integer **N** denoting the number of locks. On each of the next **N** lines, there are **N** integers, denoting the ID of each lock.

Output Format

Output a list of space separated integers denoting the ID of the locks in the order described above.

Sample Input	Sample Output	Explanation
3 1 2 3 8 9 4 7 6 5	1 2 3 4 5 6 7 8 9	First traverse the first row from left to right, then the last column from top to bottom, and the bottom row from right to left, and proceed in a spiral order.
4 1 2 4 5 0 10 11 12 6 7 8 9 13 14 15 16	1 2 4 5 12 9 16 15 14 13 6 0 10 11 8 7	First traverse the first row from left to right, then the last column from top to bottom, and the bottom row from right to left, and proceed in a spiral order.
5 1 2 3 4 5 16 17 18 19 6 15 24 25 20 7 14 23 22 21 8 13 12 11 10 9	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	As above.

6. Sharing the Spoils - Intermediate

You have discovered that the bank has a large amount of gold. While you and your team are happy with the potential spoils, you realize that you will need to divide the gold among your team. However, you cannot just distribute the gold among your team in a haphazard manner. Your subordinate, the treasurer, has come to you with several machines that take in a lump of gold, and produce gold bars of certain specific amounts.

Each machine takes in an amount of gold, and defines some specifications on the possible values of gold bars it can produce. The greedy treasurer, however, tells you that it is impossible for you to use the machines for all the gold you have collected, and says that you should distribute the gold arbitrarily. You want to prove him wrong, by showing him that you can count how many possible distributions of gold there are for any given machine specifications. He wagers half of his share of the gold if you can prove him wrong. Can you prove the treasurer wrong?

Input Format

You are given an integer **N** denoting the amount of gold you want to distribute, and an integer **C** denoting the *number of values* the machine can output for each gold bar. On the next line, you are given **C** values denoting the possible amounts the machine can manufacture each gold bar to have.

Output Format

You are required to output a single integer denoting the number of ways to distribute the gold.

Constraints

No machine will have more than 50 units of gold allocated to it.

You will not have more than 50 possible values to allocate to a gold bar.

Sample Input	Sample Output	Explanation
4 3 1 2 3	4	For $N = 4$ and $C = \{1, 2, 3\}$, we can have the following distributions: $\{1, 1, 1, 1\}$ $\{1, 1, 2\}$ $\{2, 2\}$ $\{1, 3\}$
10 4 2 5 3 6	5	For $N = 10$ and $C = \{2, 5, 3, 6\}$, we can have the following distributions: $\{2, 2, 2, 2, 2\}$, $\{2, 2, 3, 3\}$ $\{2, 2, 6\}$, $\{2, 3, 5\}$ $\{5, 5\}$

7. A Poor Pour - Intermediate

Phew, you made it out with the gold and have it secured in your safe house. The bank you stole this from has an image of a balinese dancer printed on them to make them recognizable. To avoid detection you decide to melt down the bars and remake them to destroy the images. Fortunately gold has a fairly low melting point (approx 2,000°F) so all you need is to use your welding skills to make a mold. You decide that the easiest thing to make is a box that's open on one side inside which you will add equally spaced vertical divisions. A cross section of your design filled with gold looks like this:



Unfortunately you messed up when you made the vertical division and they aren't all the same height! Remember: measure twice, cut once. Your mold looks more like this:



You decide to make due and just figure out how much gold you could possibly pour into this mold without it overflowing. You managed to get all the divisions equally spaced so you decide to write a program that, when given the heights of the divisions, will compute the maximum volume the mold can hold.

We will consider the divisions to have a width of 1 unit and there will be 5 units of space in between each division. Also, we will consider the depth of the mold to be 1 unit to make the calculations easier. In the example above if the heights of the 5 divisions were 5, 5, 3, 5, 2 then the total volume would be $5*5 + 5*5 + 2*1 + 5*5 + 2*5 = 87$. The first two spaces can be filled up to height five so we get $5*5$ twice, followed by the two units of space above the height 3 division. Then there's another division filled to height five followed by one filled to height two so we get $5*5 + 2*5$.

Input Format

You will be given an array of integers indicating the heights of all of the divisions.

Output Format

You should return an integer indicating the maximum volume of gold you can fit in the mold.

Sample Input	Sample Output	Explanation
5 5 3 5 2	87	This is the example discussed above
5 5 5 5 5	100	This is the top example of a mold with uniform height divisions

8. Image Lock - Intermediate

You are at the door to the special jewellery vault, whose lock is controlled by a mainframe deep underground. While you cannot fit into the pipes to try and reach the mainframe, you have sent a small drone to try and reconfigure it. However, the bank's programmers have anticipated such a plan, and have added a variant of an image CAPTCHA to try and prevent the use of robots! They present your drone with a sequence of images, and the drone has to decide if each one is a real photograph or a randomly generated image file. Your job now is to provide your drone with a program that can defeat the bank's image security system.

Each image will be provided as a grayscale image, with the value of each pixel ranging from 0 to 255. It is either a grayscale photograph of a cat, or each pixel is generated at random from 0 to 255.

Input Format


The first line of the input will contain two integers, **W** and **H**, representing the width and height of the image respectively. In the actual set of images on our server, each image will be between 300x300 and 700x700 pixels in size. The next **H** lines of input contain **W** integers, each between 0 and 255, each representing a single pixel in the input.

Output Format

Output a single word, either "cat", if the image is a cat, or "random", if the image is generated at random.

Extra Input Data

For testing purposes, we will be providing you with a set of images from both types of input. Use these to help you create a good program

Sample Input	Sample Output	Explanation
5 5 161 156 123 132 140 172 220 109 71 91 91 10 129 39 40 69 54 242 255 193 77 130 50 130 35	random	This is a random image. Note that these sample images are smaller than the actual ones
10 10 0 0 255 0 0 0 0 255 0 0 0 0 255 255 0 0 255 255 0 0 0 0 255 255 255 255 255 255 0 0 0 255 255 255 255 255 255 255 255 0 0 255 255 255 255 255 255 255 255 0 0 0 255 255 255 255 255 255 0 0 0 0 0 255 255 255 255 0 0 0 0 0 0 255 255 255 255 0 0 0 0 0 255 255 255 255 255 255 0 0 0 255 255 255 255 255 255 255 255 0	cat	Not a real image of a cat, just used as an example:  For actual cat images similar to those we will use, see the extra input data we provide.

9. The Laser Beam Vault - Advanced

Your team has reached a vault guarded by infrared beam alarms: if you accidentally cross a beam, you will trigger the alarms and get caught by the guards. A hacker friend of yours has successfully disabled most of the beams, but there are four left. While you cannot see where the beams are, you know some crucial information. There are exactly four beams left, forming the shape of a parallelogram. Your hacker friend has also determined a list of possible vertices of the parallelogram. Now, your job is to figure out how many possible configurations there could be for the parallelogram, to decide how risky it is to break into the vault!

Input Format

The first line of input contains a single integer **N**, indicating the number of vertices in the list provided by your hacker friend.

The next **N** lines of the input file contain two integers **x** and **y**, respectively indicating the x and y coordinates of each point.

Output Format

Output a single integer, indicating the number of parallelograms that can be formed from the set of points provided. It may be possible that no parallelograms can be formed, as your friend may have made a mistake. In that case, output 0, to indicate to him that he should check his work.

Limits

It is guaranteed that **N** is less than 2,000. Note that a brute-force approach (or even some faster approaches) is insufficient to solve this question for the largest test cases!

Sample Input	Sample Output	Explanation
5 0 0 0 2 1 1 0 4 1 5	1	Only one parallelogram can be formed, using these four points: (0, 0) (1, 1) (0, 4) (1, 5)
6 0 0 1 2 2 4 0 4 1 6 2 8	5	Five possible parallelograms can be formed from different quadruplets of points: (0, 0) (1, 2) (0, 4) (1, 6) (0, 0) (2, 4) (0, 4) (2, 8) (1, 2) (1, 6) (2, 4) (2, 8) (0, 0) (1, 6) (1, 2) (2, 8) (1, 2) (0, 4) (1, 6) (2, 4)
3 0 0 1 1 2 3	0	No parallelograms can be formed using less than 4 points.

10. Getaway Path - Advanced

Inside the main vault, there is a square $N \times N$ grid of rooms full of cash. Having spent all the trouble to get here, you need to figure out the best way to hit as many rooms as possible. Each room has a unique security level from 1 to $N \times N$, and the security system is setup such that traveling from a room with a lower security number to a higher security number will trip the alarms. You can start and end in any room you like, but while you are traveling inside the vault, you can only move to adjacent rooms (i.e. either up, down, left, or right). Figure out the maximum number of rooms you can hit in one trip!

Input format

The first line of the input file will contain a single integer N , indicating the side length of the grid. The next N lines will contain N integers each, drawing out a map of the security levels of the rooms.

Output format

Output a single integer, indicating the maximum number of rooms you can reach in one trip without going from a lower number to a higher number.

Limits

It is guaranteed that N will be less than 700.

Sample Input	Sample Output	Explanation
3 1 2 3 4 5 6 7 8 9	5	Starting in the bottom right room, you can travel at most 5 rooms (e.g. 9 - 6 - 5 - 2 - 1) in a descending sequence.
3 1 2 3 8 9 4 7 6 5	9	Starting in the middle room, you can travel in a spiral, reaching all 9 rooms in a descending sequence.
3 1 8 2 9 4 3 5 6 7	5	The best path is 7 - 6 - 4 - 3 - 2, any other descending path is shorter.

11. Cops Run into Each Other - Advanced

The most dangerous part of escape route occurs when you attempt to pass through the dreaded grid of cops located in Center City. For every block in the city there is a certain number of cops located within this block and if you passed through at any random hour, you would certainly be caught being Philadelphia's most famous bank robber. However if you pass through at lunch hour you realize that there is a weakness in the cops' defenses. In particular you send an associate to give a false tip to the cops that you are located with some rectangular region of the city. If this rectangular region has an even number of cops you won't get caught, as it is lunch hour and the cops within this region will be paired off having lunch together and nobody wants to leave his or her partner alone!

As the associate of most famous criminal in Philadelphia your job is to count the number of rectangular regions which have an even number of cops within them, as the associate always likes to know the number of choices that his boss will have.

Input format

The input format is a list of integers. The first two integers **M** and **N** indicate the width and length of an array, while the list of the next **MN** integers indicate the numbers in the array read from left to right starting in the first row and progressively working downward. Note that these integers are between 0 and 10000.

Output format

The output format is a single integer which indicates the number of rectangular subarrays in which the sum of the entries in this rectangular subarray is even.

Sample Input	Sample Output	Explanation
2 2 0 0 0 0	9	Note that this input corresponds to the array 0 0 0 0 For this input all 9 possible rectangular subarrays work.
2 3 1 1 1 1 1 1 1	10	There are 10 possible rectangular subarrays in this grid which have even sum.

12. Bugs - Advanced

While you were robbing the bank you managed to get a hold of a ledger containing all the money transfers between accounts within the bank. For these types of transactions the bank avoids actually “moving” any money between accounts by simply keeping track of who paid who how much. So the ledger contains entries of the form: account 12 sent \$10 to account 100. You also managed to snag the source code that the bank uses to implement this scheme. In the source code you discover the a bug that will let you siphon even more money from this bank. Sometimes the transactions form what is called a cycle. This is when account A sends money to B, who sends money to C, and so on until someone in the chain sends money back to A. You found that when there are an odd number of accounts in this chain it activates an error that lets you siphon a small amount of money any time someone in the cycle makes a transaction. There are a lot of accounts and transactions so you decide to write a program that will determine if a given set of transactions contains an odd length cycle.

Input Format

You will be given the number of accounts N, the number of transactions M, and a Mx2 array of integers A[i,j] where the elements of A[i,j] are in the range 0 to M-1 inclusive which indicates all the transactions. So transaction number “i” is a transfer from account A[i, 0] to account A[i, 1].

Output Format

You will output true if the transactions contain an odd length cycle and false otherwise.

Sample Input	Sample Output	Explanation
3 3 0 1 1 2 2 0	true	The first line gives the integers N and M in that order, in this case they are both 3. Account 0 sends money to account 1 which sends money to account 2 which sends it back to 0. This is a cycle of length 3 so you return true
5 6 0 1 1 2 0 2 1 3 3 4 4 0	false	The only cycle is 0-1-3-4-0 but this has length 4 which is even. Note that now account 0 sends to 2 instead of the other way around so the cycle from the last example is no longer a cycle.