

CODE-LM

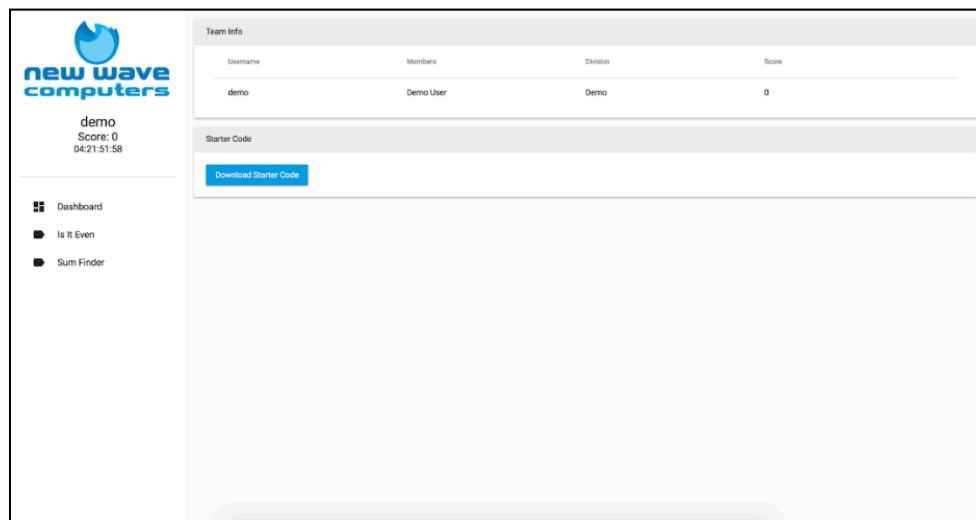
Rules: You will have until **11:30** to complete your questions. The point values of each problem are listed in the index of questions below.

The competition is managed through the **NewWaveComputers.com** website. Starter code must be downloaded from the website. Dimensions for arrays will be accounted for in the starter code. If you are programming in Java, **do not change the names of the classes in the starter code or method headers**. If you do, your code will not compile. All questions are tested and submitted through the website. You will submit your code by copying and pasting the code from your IDE into each problem's textbox on the website and then clicking the submit button.

In order to receive credit for these questions, your code must produce the correct results for all test data. This test data will not be the same as the sample data that is provided with each question in this packet, so be sure to thoroughly test your code before you submit it.

You may test your code by using the **test** button. To submit your final answer, use the **submit** button. You will only receive credit for one correct answer per problem, and you will not be penalized for an incorrect solution.

The dashboard screen contains team information and the starter code download button. The sidebar contains your score, the countdown timer, and the list of problems.



Once you have selected a question to work on, you will be taken to a problem screen which explains the problem and provides sample data. You should work on your solution in your IDE. When you are finished writing your solution, ensure that the language option is set correctly and then copy the entire contents of your file to the code editor on the website. You are encouraged to first test your solution by using the Test button. This test button will compare your results to the sample data that is visible. After testing, submit your solution by clicking the Submit button.

The screenshot shows a web interface for a coding problem titled "1. Uncrackable Encryption". At the top right, there are "Test" and "Submit" buttons. A red arrow points to the "Test" button with the text "Click here to test your code". Another red arrow points to the "Submit" button with the text "Click here once you have tested your code to submit it". The main content area is divided into two columns. The left column contains the problem description, input/output specifications, and hints. The right column contains a table with sample input and output data. Below the problem description, there is a large text area for writing code, with a red arrow pointing to it and the text "Paste your code here". At the bottom right of the code editor, there is a "Language" dropdown menu set to "Java" and a "Documentation" button.

Input	Output
coding	HoJePd
python	OpIuZq

If your solution is incorrect, you will be taken to a screen that will display any error messages that may have resulted from your submission. If you believe that your submission was incorrectly rejected, you should type a short message that describes why you think it is wrong into the dispute message box and then submit your dispute. A judge will review your dispute and give you a decision. All dispute decisions are final, but you still retain your submissions.

1. Unrackable Encryption
ID: 5ac63aba252a9a1988e3377e

Result

Error

0 points

Error

Error: Could not find or load main class UnrackableEncryption

Dispute

If you believe that your code is correct and that there is an issue with the website or problem, use the form below to dispute this submission.

Message

Send Dispute

1

This will show you the result of your code

Compilation Errors will show here

If you believe there is a problem with your submission, click here to send a dispute

If your solution is able to run, you will see a screen that displays the input and output values for each visible test case, and whether or not you got that test case correct. You can also see how many points this submission contributed to your score. Test submissions are always worth zero points, and **only fully-correct submissions award points**. Finally, if you did not receive a score of 100% and believe this to be an error, you can write an explanation in the dispute message box and submit a dispute.

1. Is It Even
ID: 5acf7d592a2d836f6ecb1b89

Result

100%

1 point

IO

Input	Correct Output	Your Output	Status
1	false	false	✓
2	true	true	✓

```

1 import java.util.Scanner;
2
3 public class IsItEven {
4     public static void main(String[] args) {
5         Scanner s = new Scanner(System.in);
6         System.out.println(s.nextInt() % 2 == 0);
7     }
8 }

```

Important Things to Remember

- You must download the starter code from NewWaveComputers.com. This starter code contains code that will read in all values from standard input and write your solution to standard output. Dimensions for arrays will be accounted for in the starter code. **You should only modify the solution method.**
- If you are using Java, do not change the class names in the starter code. If you do, your code will not compile.
- Your code should not print out anything except for the answer. Your answer must be formatted exactly like the sample data on NewWaveComputers.com or it will be marked wrong. The starter code will take care of this for you, so don't change it!
- If any member of our admin team discovers that you are using AI technologies as a resource, you will be disqualified from the competition.

Index:

Problem	Points
1: Debt Dilemma	1
2: Decisive Decisions	1
3: Phone Plunder	2
4: Offshore Obfuscation	2
5: Employee Emergency	3
6: Mechanical Madness	3
7: Wormhole Wandering	4
8: Apocalyptic Adversary	4
9: Data Decryption	5
10: Palindrome Pandemonium	5
11: Gambling Greatness	6
12: Wire Woopsie	6

Problem 1: Debt Dilemma

New Wave Computers has fallen on hard times and has been forced to declare Bankruptcy! Thanks to Mr. Swope's ill-advised decision to print trillions of dollars in NewCoins (New Wave Computer's digital currency) and many other spurious decisions, debt has skyrocketed!

You have been hired as a consultant to make as much money out of any remaining assets that New Wave Computers may hold, but, before you can do that, you'll need to figure out how much debt New Wave Computers actually has. At first, this would appear to be a very simple task. However, New Wave has kept track of all of its assets and debts in the aforementioned NewCoin instead of dollars. Additionally, it has been discovered that Mr. Swope has an off-shore bank account that uses euros.

Given the current conversion rates, write a program that returns New Wave's debt. You can assume that the debt will be greater than or equal to Mr. Swope's account balance. Report the debt as a positive number.

Conversion Rates		
Dollar (\$)	Euro (€)	NewCoin (N)
1	0.917	20

Input:

double currentDebt = a double representation of New Wave's current debt, in NewCoins

double balance = a double representation of Mr. Swope's bank account balance, in euros

Output:

double totalDebt = New Wave's total debt after including Mr. Swope's account balance

Sample Data:

Input	Output	Explanation
double currentDebt = 400 double balance = 10	double totalDebt = 9.10	<p>The current debt is 400 NewCoins; to convert this to dollars, you would divide it by 20, so 20 dollars of current debt remain.</p> <p>The balance is 10 euros; to convert this to dollars you would multiply it by 1.09, so 10.9 dollars is the balance.</p> <p>20 dollars of debt - 10.9 dollars of balance = 9.1 dollars of total debt.</p>
double currentDebt = 100 double balance = 2	double totalDebt = 2.82	<p>The current debt is 100 NewCoins; to convert this to dollars, you would divide it by 20, so 5 dollars is the current debt.</p> <p>The balance is 2 euros; to convert this to dollars, you would multiply it by 1.09, so 2.18 dollars is the balance.</p> <p>5 dollars - 2.18 dollars = 2.82 dollars of debt</p>
double currentDebt = 4000 double balance = 20	double totalDebt = 178.2	<p>The current debt is 4000 NewCoins; to convert this to dollars, you would divide it by 20, so 200 dollars is the current debt.</p> <p>The balance is 20 euros; to convert this to dollars, you would multiply it by 1.09, so 21.8 dollars is the balance.</p> <p>200 dollars - 21.8 dollars = 178.2 dollars of debt</p>

Problem 2: Decisive Decisions

Now that the debt has been calculated, you'll need to go through each department at New Wave Computers to determine if the department should be liquidated, exempt, or laundered. You'll be given three attributes for each department: annual sales, debt and equity.

If a department has a debt-to-equity ratio of 80% or more, it should be laundered. If its debt-to-equity ratio is less than 80% and its sales to debt ratio is greater than 50%, it should be exempt. Otherwise, the department should be liquidated.

Given a department's annual sales, debt, and equity, write a program that returns "liquidated", "exempt", or "laundered".

Input:

int annualSales = the department's annual sales
int debt = the department's debt
int equity = the department's equity

Output:

String action = the action that should be taken against the department

Sample Data:

Input	Output	Explanation
int annualSales = 10000 int debt = 30000 int equity = 20000	String action = "laundered"	30000/20000 > 80%, so the department should be laundered
int annualSales = 100000 int debt = 20000 int equity = 30000	String action = "exempt"	20000/30000 <= 80% and 100000/20000 > 50%, so the department should be exempt
int annualSales = 1000 int debt = 20000 int equity = 30000	String action = "liquidated"	20000/30000 <= 80% and 1000/20000 < 50%, so the department should be liquidated

Problem 3: Offshore Obfuscation

Hoping to hide money from the IRS, Mr. Swope often keeps money in offshore bank accounts. In an attempt to ensure account secrecy, his team of financial advisors have encoded his financial data with a rather strange pattern. Apparently, the balance of each account is listed inside of a string of numbers and characters. However, only the numbers count towards the balance. Consecutive numbers, which aren't separated by characters, should be considered a single number.

For instance, given the input "a2d83D44fAdsZ3dd38j," the initial balance would be $2 + 83 + 44 + 3 + 38 = 170$.

Given a string of Mr. Swope's financial data, write a program that returns his balance.

Input:

String pass = a string of Mr. Swope's encoded financial data

Output:

int balance = an int of the decoded balance

Sample Data:

Input	Output	Explanation
String pass = "a343S567hhFd5ygIu17"	int balance = 932	$343 + 567 + 5 + 17 = 932$
String pass = "NewWave27kVjlLkJ2kk60K8j7"	int balance = 104	$27 + 2 + 60 + 8 + 7 = 104$
String pass = "iLuvC0mPuter5"	int balance = 5	$0 + 5 = 5$

Problem 4: Phone Plunder

You've determined that New Wave's mobile department should in-fact be liquidated. Their once groundbreaking line-up of newPhones, Nixels, and Nebulas did not sell as expected, primarily because the phones' batteries had been known to charge slowly and spontaneously combust. New Wave has an extensive, but flawed inventory that lists the location, phone type, and quantities in an array of arrays. For example, the list

```
[[Springfield, newPhone, 323], [Ardmore, Nixel, 5000], [Norristown, Nebula, 9]]
```

Would indicate a stock of 323 newPhones in Springfield, 5000 Nixels in Ardmore, and 9 Nebulas in Norristown.

As was previously stated, this inventory does however have several flaws. The first flaw that you may encounter is repeated, and perhaps inconsistent, phone quantities at a particular store. When this occurs, the phones should only be counted once, and you should assume that the entree with the lowest stock is more accurate than an entree that has a higher quantity. For example, you may encounter the two following entries in the array of items...

```
[Springfield, newPhone, 323], [Springfield, newPhone, 5000]
```

In this case you should only count 323 newPhones at the Springfield store.

Given a 2D array of the inventory, write a program that returns the total amount of newPhones, Nixels and Nebulas.

(Hint: `Integer.parseInt()` or `Integer.valueOf()`)

Input:

String[][] inventoryFlawed = a 2D array of phone type and their quantities

Output:

int totalPhones = an int of the total number newPhones, Nixels, and Nebulas

Sample Data:

Input	Output	Explanation
String[][] inventoryFlawed = [[Ardmore, newPhones, 100], [Springfield, Nebulas, 200], [Norristown, Nixels, 400]]	int totalPhones = 700	There are 400 Nixels, 200 Nebulas, and 100 newPhones, which add up to 700.
String[][] inventoryFlawed = [[Springfield, newPhones, 100], [Delco, Nebulas, 200], [Delco, Nixels, 400], [Springfield, newPhones, 50]]	int totalPhones = 650	There are two entries of newPhones at Springfield; therefore, the lower quantity of 50 is used.
String[][] inventoryFlawed = [[Ardmore, newPhones, 100], [Delco, Nebulas, 200], [Delco, Nebulas, 400], [Ardmore, newPhones, 50]]	int totalPhones = 250	There are duplicate entries of both newPhones and Nebulas at a particular location; therefore, the lower quantities are used.

Problem 5: Employee Emergency

In such hard times, tensions are running high in the offices of New Wave Computers. The company is facing financial difficulties, and employees are protesting against management decisions. Many are dissatisfied with reduced wages and mass layoffs. Some have even threatened to sue! The Human Resources department has become the center of the conflict, with employees demanding attention to be given to their grievances.

In an effort to peacefully protest, employees have agreed to wait in line for help. But, their patience is limited. Each employee is only willing to wait behind a certain number of others. An employee will not join the line if they have to wait behind too many people. You need to best organize the line so that HR can help as many employees as possible.

For example, take the four employees in the array [2, 0, 0, 4]. The first employee is willing to wait behind two people, the second and third are not willing to wait at all, and the fourth is willing to wait behind four people. At best, you can organize a line to help three employees. You would put one of the employees who cannot wait behind anyone first in line, then the employee who can wait behind 2 people, and finally the employee who has the patience to wait behind 4 people. However, it is impossible to get both employees who are not willing to wait behind anyone to join the line and get help.

Given an array of employee "patience", write a program that returns the most amount of employees you can convince to join the line.

Input:

int[] waitTimes = an array of how many people each employee is willing to wait behind

Output:

int max = an int of the maximum number of employees that can be convinced to join the line

Sample Data:

Input	Output	Explanation
<code>int[] waitTimes = [0, 1, 2, 3, 4]</code>	<code>int max = 5</code>	If you attend to the employees in the order they are given in the array, you can attend to all the employees.
<code>int[] waitTimes = [2, 1, 0, 3, 1, 1]</code>	<code>int max = 4</code>	The max is 4 because if you attend no matter how the line is organized, two employees will have their wait times surpassed.
<code>int[] waitTimes = [3, 1, 1, 6, 5, 2, 4]</code>	<code>int max = 7</code>	Even though there are two employees who are only willing to wait behind 1 colleague, all of the employees can be attended to if they are ordered [1, 1, 2, 3, 4, 5, 6]

Problem 6: Mechanical Madness

To make the most out of the failed Mobile Department, New Wave Computers has collected all the broken newPhones, Nebulas, and Nixels in their inventory. The company hopes that if you can fix the phones, they could still be sold for cheap. As a consultant for New Wave Computers, you have hired several mechanics who can fix the phones. Given the number of mechanics and a list representing the broken phones with the time it takes to fix each phone in minutes, determine the total minutes needed to fix all of the faulty hardware. Each mechanic begins working at time zero. Phones are fixed in the order they appear in the array beginning at index 0. When a mechanic finishes fixing a phone, they immediately begin fixing the next phone in line. If more than one mechanic finishes at the same time, it does not matter how the next phones are assigned.

For example, if the line of phones is [1, 3, 4, 5] and you have hired two mechanics, you would return 8. At time zero, the first mechanic would start repairing the first phone and the second mechanic would start repairing the second phone. After 1 minute, the first mechanic would finish and start repairing the third phone. After three minutes, the second mechanic would finish and start repairing the fourth and final phone. The first mechanic finishes again at 5 minutes, but the second mechanic will only finish after 8 minutes. Therefore, you would return 8.

Given an array of phone repair times and number of mechanics, write a program that returns the time needed to finish repairing all phones.

Input:

int[] phones = an array of how long it takes to fix each phone
int mechanics = the number of mechanics available

Output:

int minTime = an int of the minimum time it takes to fix all phones

Sample Data:

Input	Output	Explanation
<code>int[] line = [1, 2, 3]</code> <code>int mechanics = 3</code>	<code>int minTime = 3</code>	Each mechanic begins to fix one of the three phones on the assembly line. After 3 minutes, the last mechanic will finish the last phone.
<code>int[] line = [1, 4, 5, 5]</code> <code>int mechanics = 3</code>	<code>int minTime = 6</code>	Similar to the example above, the three mechanics begin to fix the first three phones. After the first mechanic finishes the first phone, they begin to work on the last phone. That mechanic will finish last for a total time of 6 minutes.
<code>int[] line = [1, 1, 1, 1]</code> <code>Int mechanics = 1</code>	<code>Int minTime = 4</code>	The single mechanic will have to fix each phone one by one, which will take 4 minutes in total.

Problem 7: Wormhole Wandering

New Wave Computers was once on the brink of stardom. Years ago, the “Time Travel Development Unit” made headlines across the nation. New Wave Computers invested millions of dollars into the time traveling technologies that would place the company on the Mount Rushmore of the tech world. However, the product release could not have been less successful. Several users of the product became lost in time, never returning back to the present. New Wave Computers argues that the users did not use the time machines properly and got trapped in time. However, the families of the users say that we designed faulty navigation systems (Blasphemy!). These families have sued us for our “supposed” wrongs and we have countersued them for defamation (trying to damage our reputation). We need you to code the navigation system to help us win the lawsuit, and remove some of our debt.

In order to complete this task, create code that when given a square maze (representing a time travel wormhole) in the form of a 2D array of characters, where X represents obstacles, periods are open space, and O’s are portals, outputs the number of steps it takes to get from S to F without backtracking. Each maze will have exactly one optimal solution. When you encounter a portal, you have two options: treat it as an open space or use the portal to teleport to another portal. Using a portal to teleport counts as one step.

Given a 2D array, write a program that returns the minimum number of steps to get from start (S) to finish (F).

Input:

char[][] maze = a 2D array of the maze, guaranteed to be a **square maze** (N rows and N columns, where N is an integer)

Output:

int steps = the number of steps needed to travel from S to F

Sample Data:

Input	Output	Explanation
<pre>char[][] maze = ["XXXX", "X..S", "X.XX", "F.XX"]</pre>	<pre>int steps = 5</pre>	The optimal path to go from the S to F presented in the 2D array of characters requires 5 steps: left, left, down, down, left.
<pre>char[][] maze = ["XXXXXXXXXX", "XS..X....X.", "X.XXX.XXX.X", "X....X..X.X", "X.XXXXX.XXX", "X....O..X.X", "XXXXXXXX.XOX", "X.O...X...X", "X.XXX.XXX.X", "X..X....OFX", "XXXXXXXXXXXX"]</pre>	<pre>int steps = 10</pre>	Going to the highest portal first, then traveling to the portal next to the finish is the optimal path. You travel 4 down, 4 right, 1 step to teleport, then 1 more right. $4+4+1+1 = 10$ steps.
<pre>char[][] maze = ["XXSXX", "XX.XX", "XXOXX", "XX.XX", "XXF.O"]</pre>	<pre>int steps = 4</pre>	It is more optimal to walk across the portal than to take it. 4 down = 4 steps.

Problem 8: Apocalyptic Adversary

In a desperate attempt to avoid bankruptcy, New Wave Computers has decided to unleash a zombie virus upon the world. Anything is better than debt, right? However, this plan kind of backfired. All is not lost though. In fact, New Wave Computers has a secret collection of weapons! The damage of each weapon will be provided in an array.

The zombies affected by the virus are not normal. They can only be injured if the damage dealt to them is exactly equal to their health. The health of each zombie will also be provided in an array

(Look at the test cases to understand this process better).

Weapons are not reusable. Be careful which you select to defeat each zombie. Weapons can be combined to combat the zombies. You must determine if the New Wave Computers can defend itself from the numerous zombies. Return true if New Wave Computers can defeat all the zombies and false if otherwise.

Given an array of weapons and an array of zombies' health, write a program that returns true or false.

Input:

int[] weapons = list of weapons

int[] zombies = list of zombies' health

Output:

boolean success = true if zombies can be defeated, false if otherwise

Sample Data:

Input	Output	Explanation
<code>int[] weapons = [2,3,1]</code> <code>int[] zombies = [3,3]</code>	<code>boolean success =</code> True	The first zombie can be defeated with a combination of the 2 and 1 damage weapons. Afterwards, the second zombie can be defeated using the 3 damage weapon.
<code>int[] weapons = [2,2,2]</code> <code>int[] zombies = [3,3]</code>	<code>boolean success =</code> False	No exact damage can be done to the zombies using the weapons available. Therefore, they cannot be defeated.
<code>int[] weapons = [4,5,1,</code> 3, 2] <code>int[] zombies = [2, 9, 4]</code>	<code>boolean success =</code> True	Although the zombie with 4 health could be defeated using either the 4 damage weapon or a combination of the 1 and 3 damage weapons, you must use the 1 and 3 damage weapons so that you can use the 4 and 5 damage weapons to take down the 9 health zombie.

Problem 9: Data Decryption

As previously mentioned, Mr. Swope contains large sums of money in an offshore Bank Account, hidden from the IRS. In these urgent times, this money needs to be accessed to reduce the debt of New Wave Computers, an action that Swope has put off for a long time. This is not due to his lack of generosity, but rather his extreme caution (paranoia). For whatever reason, he decided to create a scrambler that affected the password of his account, and now has no clue how to access the funds.

The original New Wave Computers password consisted of letters from 'a' to 'z'.

1. The password is the same length as the original password (≥ 1 letter).
2. The password contains the same number of each character that the original password contains (the new password is an "anagram" of the original).
3. No letter in the new password ends up in the same position as it was in the original password. If the password contains multiple of the same letter, then no letter can end in the position of an identical letter in the original password.

The examples below illustrate a new password that is valid and invalid for the original password: "password"

Valid	Invalid
"dsawpros"	"psawsdor" ("p" appears in the same position as it does in the original)
"soprawds"	"ropsadws" ("s" appears in the one of the original positions)

Given a string of the password, write a program that will return the number of possible new passwords.

Input:

string password = the original password; length will be at most 25 characters

Output:

int possibilities = this outcome should be the number of possible passwords

Sample Data:

Input	Output	Explanation
<code>string password = "hack"</code>	<code>int possibilities = 9</code>	<p>The input "hack" has 1 'h', 1 'a', 1 'c', and 1 'k'. The new password must contain the same number of each of these letters.</p> <p>The 'h' cannot be in the 1st position in the new password. The 'a' cannot be in the 2nd position. The 'c' cannot be in the 3rd position. And the 'k' cannot be in the 4th position.</p>
<code>string password = "eagles"</code>	<code>int possibilities = 84</code>	<p>The input "eagles" has 2 'e's, 1 'a', 1 'g', 1 'l', and 1 's'. The new password must contain the same number of each of these letters.</p> <p>The 'e' cannot be in the 1st OR 5th position in the new password. The 'a' cannot be in the 2nd position. The 'g' cannot be in the 3rd position. The 'l' cannot be in the 4th position. And the 's' cannot be in the 6th.</p>
<code>string password = "anagram"</code>	<code>int possibilities = 72</code>	<p>The input "anagram" has 3 'a's, 1 'n', 1 'g', 1 'r', and 1 'm'. The new password must contain the same number of each of these letters.</p> <p>The 'a' cannot be in the 1st OR 3rd OR 6th position in the new password. The 'n' cannot be in the 2nd position. The 'g' cannot be in the 4th position. The 'r' cannot be in the 5th position. And the 'm' cannot be in the 7th.</p>

Problem 10: Palindrome Pandemonium

Hopefully, you have helped to repair the faulty navigation systems of our time machines. Before we seal the win in the lawsuit, we need to show that the hardware for the machines is like everything else at New Wave Computers: perfect. The maintenance team knows how to identify all of the faulty hardware, but they need to know how many parts of the machine are broken.

You are given an integer N representing the dimensions of the cube and a square 2D array of dimensions $N \times (N \times N)$ to represent the $N \times N \times N$ cube. Each 1D array in the 2D array represents a layer in the cube (1st is 1st layer, 2nd is 2nd layer, etc). This array has $N \times N$ numbers, and you should convert this to a 2D array to represent that square layer.

(An example of this process is shown below)

You will also be given a point with an x , y , and z index and you need to determine how many FULL lines containing the point, in any direction, can make a palindrome number.

(A FULL line is the largest line you can make in a given direction with that point.)

These palindrome numbers represent all of the broken parts of the machine. These directions include horizontally, vertically, and diagonally across one layer, depth-wise, and multiple layers of the cube; palindromes are only valid following one of these directions in a straight line.

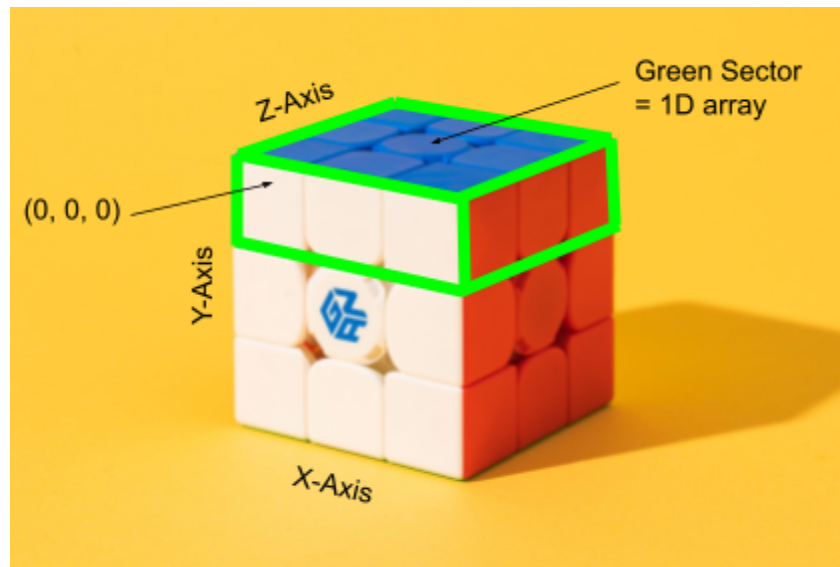
Given a 2D array, write a program that returns the number of palindrome numbers found throughout the cube.

A palindrome number is a number that reads the same backwards as it does forwards. Some examples of palindrome numbers are 12321, 11, and 565. The numbers added to the arrays are assigned randomly.

For example, take a $2 \times 2 \times 2$ cube in 2D format: `[[1,2,3,4], [5,6,7,8]]`

In 3D format: `[[[1,2],[3,4]], [[5,6],[7,8]]]`

Start point and direction diagram



Input:

int N = The dimensions of the cube

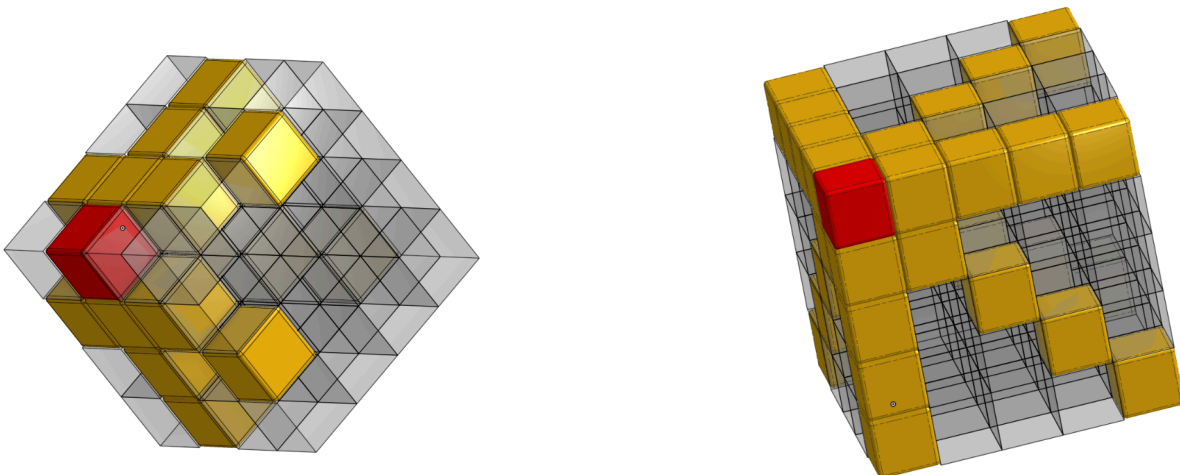
int [[[cube = the 2D array which you should convert into the 3D cube in which the palindromes need to be found within

int [] point = the array containing an x, y, and z index for the point's location; every line to detect palindrome numbers should contain this point

Output:

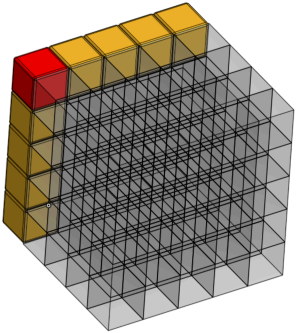
int numberOfPalindromes = the number of palindrome numbers which have more than one digit in different directions around the cube.

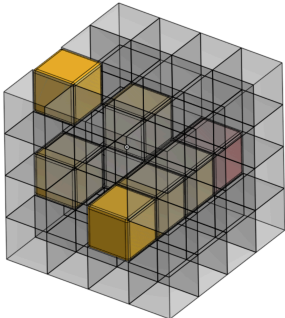
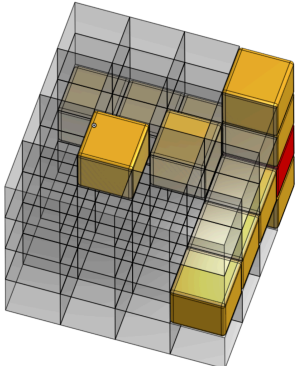
Images below show the 3D array represented by cubes. The red cube is the point. The yellow cubes represent all the possible palindrome numbers coming off from that point.



Sample Data:

(Matrix and explanation may be incorrect, focus on input/output)

Input	Output	Matrix	Explanation
<pre> int N = 5 int[][][] cube = [[1,2,3,2,1,3,4,3, 6,4,3,5,5,6,5,4,6, 8,4,3,5,2,7,9,2], [1,3,0,4,3,2,2,6,1 ,2,3,4,6,9,7,3,7,6 ,4,9,4,5,8,6,3], [1,9,5,4,2,1,2,3,2 ,1,4,5,3,6,7,1,8,0 ,5,6,2,8,9,0,0], [1,0,8,9,0,3,5,7,8 ,9,6,2,1,3,5,7,4,3 ,2,2,0,0,9,8,5], [1,3,4,4,4,0,9,8,7 ,5,1,1,2,2,0,2,4,4 ,5,4,3,4,6,9,1]] int[] point = [0,0,0] </pre>	<pre> int numberOfPalindromes = 3 </pre>	$A = \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 3 & 4 & 3 & 6 & 4 \\ 3 & 5 & 5 & 6 & 5 \\ 4 & 6 & 8 & 4 & 3 \\ 5 & 2 & 7 & 9 & 2 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 3 & 0 & 4 & 3 \\ 2 & 2 & 6 & 1 & 2 \\ 3 & 4 & 6 & 9 & 7 \\ 3 & 7 & 6 & 4 & 9 \\ 4 & 5 & 8 & 6 & 3 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 9 & 5 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \\ 4 & 5 & 3 & 6 & 7 \\ 1 & 8 & 0 & 5 & 6 \\ 2 & 8 & 9 & 0 & 0 \end{bmatrix}$ $D = \begin{bmatrix} 1 & 0 & 8 & 9 & 0 \\ 3 & 5 & 7 & 8 & 9 \\ 6 & 2 & 1 & 3 & 5 \\ 7 & 4 & 3 & 2 & 2 \\ 0 & 0 & 9 & 8 & 5 \end{bmatrix}$ $E = \begin{bmatrix} 1 & 3 & 4 & 4 & 4 \\ 0 & 9 & 8 & 7 & 5 \\ 1 & 1 & 2 & 2 & 0 \\ 2 & 4 & 4 & 5 & 4 \\ 3 & 4 & 6 & 9 & 1 \end{bmatrix}$	<p>We are starting at point 0, 0, 0, which is the number 1 in the top left corner of matrix A. From this point, with matrix C, we can see 2 palindrome numbers: one along the y-axis and one along the x-axis.</p> 

<pre> int N = 4 int[][][] cube = [[2,1,5,4,1,4,3,5, 6,9,6,6,5,1,6,7], [8,4,4,4,9,9,1,9,3 ,7,3,1,9,4,2,1], [1,4,6,4,3,1,7,8,6 ,2,7,7,3,1,6,8], [9,1,4,6,2,1,5,1,7 ,7,2,1,7,5,6,7]] int[] point = [2,0,2] </pre>	<pre> int numberOfPalindromes = 4 </pre>	$A = \begin{bmatrix} 2 & 1 & 5 & 4 \\ 1 & 4 & 3 & 5 \\ 6 & 9 & 6 & 6 \\ 5 & 1 & 6 & 7 \end{bmatrix}$ <hr/> $B = \begin{bmatrix} 8 & 4 & 4 & 4 \\ 9 & 9 & 1 & 9 \\ 3 & 7 & 3 & 1 \\ 9 & 4 & 2 & 1 \end{bmatrix}$ <hr/> $C = \begin{bmatrix} 1 & 4 & 6 & 4 \\ 3 & 1 & 7 & 8 \\ 6 & 2 & 7 & 7 \\ 3 & 1 & 6 & 8 \end{bmatrix}$ <hr/> $D = \begin{bmatrix} 9 & 1 & 4 & 6 \\ 2 & 1 & 5 & 1 \\ 7 & 7 & 2 & 1 \\ 7 & 5 & 6 & 7 \end{bmatrix}$	<p>We are starting at point 2, 0, 2 which is the 6th number in the 3rd column and top row of matrix C. From this point with matrix C we can see 3 palindrome numbers: one along the z-axis, one diagonal on a 2d plane, and one diagonal on a 3d plane.</p> 
<pre> int N = int[][][] cube = [[5,6,1,2,4,6,7,7, 2,8,9,3,4,5,7,1], [6,2,3,8,4,2,5,9,0 ,5,2,8,8,2,0,1], [8,7,7,8,9,0,1,4,6 ,8,6,4,9,0,9,8], [3,0,2,2,7,3,4,2,2 ,9,5,7,2,3,9,7]] int[] point = [2,0,3] </pre>	<pre> int numberOfPalindromes = 4 **Change the picture since it's wrong? </pre>	$A = \begin{bmatrix} 5 & 6 & 1 & 2 \\ 4 & 6 & 7 & 7 \\ 2 & 8 & 9 & 3 \\ 4 & 5 & 7 & 1 \end{bmatrix}$ <hr/> $B = \begin{bmatrix} 6 & 2 & 3 & 8 \\ 4 & 2 & 5 & 9 \\ 0 & 5 & 2 & 8 \\ 8 & 2 & 0 & 1 \end{bmatrix}$ <hr/> $C = \begin{bmatrix} 8 & 7 & 7 & 8 \\ 9 & 0 & 1 & 4 \\ 6 & 8 & 6 & 4 \\ 9 & 0 & 9 & 8 \end{bmatrix}$ <hr/> $D = \begin{bmatrix} 3 & 0 & 2 & 2 \\ 7 & 3 & 4 & 2 \\ 2 & 9 & 5 & 7 \\ 2 & 3 & 9 & 7 \end{bmatrix}$	<p>We are starting at point 2,0,3 which is the number 8 in the top right hand corner of matrix C. From this point, with matrix A, we can see 4 palindrome numbers: one along the z-axis, one along the x-axis, one along the y-axis, and one diagonal to the top in the 3-D plane.</p> 

Problem 11: Gambling Greatness

Due to poor management, New Wave Computers has been losing a lot of money in our casinos. We have, however, determined that the business is salvageable and does not need to be liquidated. In order to start making a profit off our casinos, you must win money back from the New Wave gamblers, who cannot resist a good "game". Ironically, these gamblers have been suspected of counting cards.

They challenge you to a simple two-player game. Given an array full of cards and their associated values, determine if you can win the game and, therefore, the gambler's money. Each player takes turns picking a card from either the start or end of the array and adding that card's value to their score. The game is played until all cards are picked. Whoever has the highest score at the end wins. It is assumed that you get to go first and both players play optimally. Return a boolean - true if you will win/tie and false if the gambler will win.

For example, if you were given the array [1, 2, 2], you should return true because on your first turn you could pick either the 1 or 2. Afterwards, the gambler could pick either the 1 or 2 as well depending on your pick. Regardless, you will be able to pick the final card and any order of "picking" will result in you having a higher score than the gambler.

Given an array of cards, write a program that returns true or false.

Input:

int[] cards = an array of the cards in the game

Output:

boolean canWin = whether or not you can win against the gambler

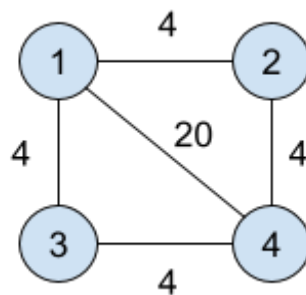
Sample Data:

Input	Output	Explanation
<code>int[] cards = [1,5,2]</code>	<code>boolean canWin = false</code>	Initially, you can choose between 1 and 2. If you choose 2 (or 1), then the gambler can choose from 1 (or 2) and 5. If the gambler chooses 5, then player 1 will be left with 1 (or 2). So, your final score is $1 + 2 = 3$, and the gambler's final score is 5. Hence, you will never be the winner and you should return false.
<code>int[] cards = [1,5,233,7]</code>	<code>boolean canWin = true</code>	You first choose 1. Then the gambler has to choose between 5 and 7. No matter which number the gambler chooses, you can choose 233. In the end, you have a higher score (234) than the gambler (12), so you should return true representing that you can win.
<code>int[] cards = [10, 2]</code>	<code>boolean canWin = true</code>	You first choose 10. Then the gambler chooses 2. You have a higher score and should return true.

Problem 12: Wire Woopsie

In previous years, New Wave Computers has attempted to launch a prestigious line of New Wave Rockets. However, there always seemed to be an issue with the wire box. Scientists think that unnecessary wiring is the issue. If you can minimize the length of wiring needed for the wire box, New Wave Computers will be able launch their rockets and generate revenue from stock investors.

A wire box is filled with nodes. Nodes are connected to each other via wires of different lengths. To ensure that the wire box works, all nodes must be connected to each other. You must determine the minimum length of wire needed to connect all nodes or whether it is impossible to have all nodes (wire boxes) connected with the given wire.



In the figure above, the minimum length of wire needed to all nodes is 12. You can remove the wire connecting nodes 1 to 4 and one of the wires of length 4.

Given the number of nodes and a 2D array of node connections, write a program that returns the minimum length of wire needed to connect all nodes.

Input:

int n = the number of nodes

int[][] connections = a 2D array, with each int array containing values:

[a,b,x] (a = node, b = node, x = the length of the wire connecting them). The values of a and b range from 1 to n, and it is guaranteed that a != b.

Output:

int minLength = the minimum sum of lengths of wires required to connect all the wire boxes, output -1 if it is **not possible** to connect all the wire boxes together with the given wires.

Sample Data:

Input	Output	Explanation
<pre>int n = 3 int[][] connections = [[1, 2]]</pre>	<pre>int minLength = -1</pre>	There are three nodes, but only two of them are connected. Therefore, the nodes are not fully connected.
<pre>int n = 4 int[][] connections = [[1, 2, 3], [2, 4, 5], [2, 3, 6], [4, 3, 4], [1, 4, 7]]</pre>	<pre>int minLength = 12</pre>	Using the wires connecting 1 to 2, 2 to 4, and 4 to 3, you fully connect the wire boxes with a sum of $3 + 5 + 4 = 12$. It can be shown that you can not get a smaller sum of wires which fully connects the nodes.
<pre>int n = 5 int[][] connections = [[1, 2, 1], [2, 3, 1], [4, 5, 1]]</pre>	<pre>int minLength = -1</pre>	No wires can connect group [1,2,3] to group [4,5], so the wire boxes can not become fully connected.