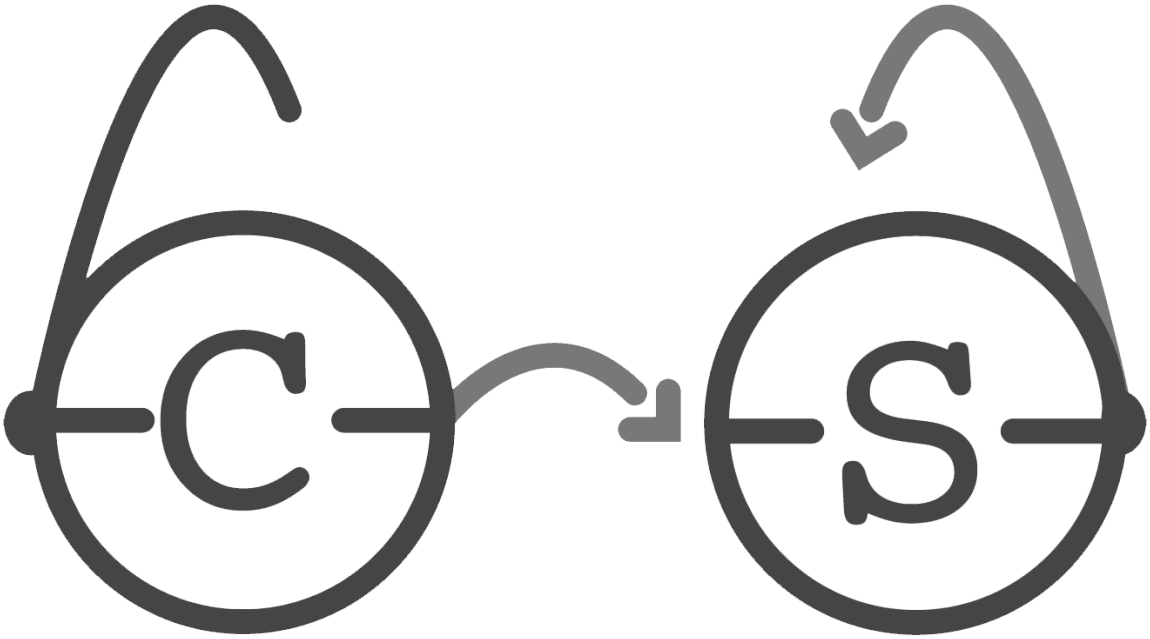
Contest Problems
Philadelphia Classic, Fall 2019
Hosted by the Computer Science Society
University of Pennsylvania



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. Snap Save

Evil Wizard wants to perform a spell that would kill 90% of the life on Earth instantly. In order to perform this spell, he must be at the top of a mountain. Evil Wizard will attempt to quickly reach the top of the closest mountain possible, traveling at a horizontal rate of one mile per second. The peak of a mountain is a location that has height greater or equal to all heights around it. Fortunately, Luke the Amazing Wizard can instantly take away Evil Wizard's powers with a single snap. Luke, aware of Evil Wizard's plans, wants to save the world at the last possible second to look cool. Using the topographic data, help Luke determine how long he can wait before snapping his fingers to prevent Evil Wizard from killing 90% of the life on Earth the moment Evil Wizard reaches the first peak of a mountain.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`findPeak(heights)`

- **`heights`**: an integer array that represents the heights in the direction Evil Wizard will travel, starting from the leftmost location and going right.

Your function should return an integer representing the number of seconds Luke can wait before he snaps his fingers.

Constraints

$2 \leq |heights| \leq 8000$

Sample Input	Sample Output	Explanation
`findPeak([20, 10, 35])`	`0`	Luke can wait 0 seconds, as Evil Wizard starts on a peak of 20 (20 ≥ 10)
`findPeak([13, 52, 75, -8, 27])`	`2`	Luke can wait 2 seconds, as Evil Wizard will reach a peak height of 75 in 2 seconds (75 ≥ 52 and 75 ≥ -8)

2. Flying Quiz

During lunch time, your friend tells you that there will be a broomflying pop quiz this afternoon. In the Magical Learning Gym, there are n consecutives stones in a row and each with a different height. To pass the quiz, you need to successfully (and gracefully) fly from a stone to the consecutive one. Since the smaller the height difference is, the higher chance of you passing the quiz is, thus, as a newbie, you want to find the minimum difference between two consecutive stones.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`findMinDiff(numbers)`

- **`numbers:`** an array of integers (yes, in magical school the stone can definitely have negative height)

Your function should return an integer that represents the minimum difference. Our test cases make sure that the answer will not exceed the range of integers.

Constraints

$2 \leq |numbers| \leq 10^8$

$-2147483648 < numbers[i] < 21474836347$ (*aka all integers*)

Sample Input	Sample Output	Explanation
`findMinDiff([1, 34, 4, 6, 14, 2])`	`2`	The minimum difference is $\|4-6\| = 2$
`findMinDiff([23, 15, 46, 75, 6, 74, 100])`	`8`	The minimum difference is $\|23-15\| = 8$

3. Bit Spill

Merlin plans to stop his arch-rival Alatar's evil plans by spilling bit-serum on Alatar's spell-card, a table of bits storing his most secret formulas. Merlin's bit-serum has the unique property that once dropped on a bit, it will spread and toggle any neighboring bit originally in the same state. Merlin only has limited amounts of serum, and he wants to find the pattern of droplets that will maximize the damage to Alatar's secret formulas. Given that Merlin knows only the initial state of the card, help him simulate the damage spilling a certain constellation of drops will do to Alatar's spell-card.

Implementation Details

Implement the following function. It will be called by the grader multiple times for each test.

`bitSpill(x, y, m)`

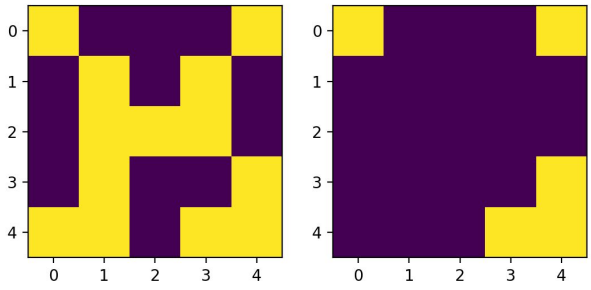
- **`m`**: a 2D matrix of 0's and 1's
- **x**, **y:** row and column coordinates of where the droplets will be spilled, respectively.

Your function does not need to return anything, as **m** is a reference to the matrix, meaning that your changes will be updated across the code.

Constraints

$2 \leq s \leq 1000$, where s is the side length of the 2D array

Sample Input	Sample Output	Input/Output (1: light color, 0: dark color)

`bitSpill(2, 1,`
`[[1, 0, 0, 0, 1],`
`[0, 1, 0, 1, 0],`
`[0, 1, 1, 1, 0],`
`[0, 1, 0, 0, 1],`
`[1, 1, 0, 1, 1]])`
	`[[1, 0, 0, 0, 1],`
	`[0, 0, 0, 0, 0],`
	`[0, 0, 0, 0, 0],`
	`[0, 0, 0, 0, 1],`
	`[0, 0, 0, 1, 1]]`
		

4. Roman Numeral

You are studying for your next test on Muggle history when you come across some strange number system. It's created by some people called the Romans, apparently. Once you learn how to decipher them you come up with a brilliant idea: wouldn't it be cool if you could come up with a spell to turn normal numbers into Roman numerals?! You could show it to your friends; you could be popular! Probably not, but it's a neat idea and you set out to write a program to double check that your spell actually works (because the computer never lies!)
Below is a chart that describes how much each roman numeral symbol is valued in decimal number form, along with rules to follow for roman numerals:

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1,000

1. Repeating a numeral up to three times represents addition of the number. For example, III represents 1 + 1 + 1 = 3. Only I, X, C, and M can be repeated; V, L, and D cannot be, and there is no need to do so.
2. Writing numerals that decrease from left to right represents addition of the numbers. For example, LX represents 50 + 10 = 60 and XVI represents 10 + 5 + 1 = 16.
3. To write a number that otherwise would take repeating of a numeral four or more times, there is a subtraction rule. Writing a smaller numeral to the left of a larger numeral represents subtraction. For example, IV represents 5 - 1 = 4 and IX represents 10 - 1 = 9. To avoid ambiguity, the only pairs of numerals that use this subtraction rule are listed below (continued on next page!):

4. Roman Numeral

Roman Numeral	Arabic Equivalent
IV	4 = 5 - 1
IX	9 = 10 - 1
XL	40 = 50 - 10
XC	90 = 100 - 10
CD	400 = 500 - 100
CM	900 = 1000 - 100

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`solve(dec)`

- **dec**: a normal decimal number that needs to be converted to Roman numeral form

Your function should return a String that is the Roman numeral form of dec

Constraints

dec $\leq$3,999

Sample Input	Sample Output	Explanation
`solve(435)`	`CDXXXV`	400 maps to CD (follows the subtraction rule), 30 maps to XXX and 5 maps to V
`solve(1904)`	`MCMIV`	1000 maps to M, 900 maps to CM (follows the subtraction rule), 4 maps to IV

5. Studying Magic: Practice!

Whether you are a beginner or an advanced magic practitioner, practice makes perfect. In PClassic School of Magical Learning, each classroom offers tests for one specific ability you can practice and each test is associated with a difficulty level. In order to achieve optimal performance, you should choose three different tests with strictly increasing level of difficulty to practice. That is, in classroom Broomflying, if tests Broomflying[i] < Broomflying[j] < Broomflying[k], then (i, j, k) is considered as a valid choice. Your task is to help the students figure out how many valid choices there are for the classroom they are visiting so that they can try out different combinations.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`findTriples(A[])`

- **`A[]`**: an array of integers containing the difficulty levels of all abilities offered in this classroom

Your function should return an integer that is the number of valid choices in classroom A mod 1000000007. The mod part is just to make sure the result still lies in integer value range.

Constraints

$3 \leq size\ of\ A \leq 10^6$

$-2147483648 < A[i] < 21474836347$ ($aka\ are\ all\ integers$)

(cont. on next page)

5. Studying Magic: Practice!

Sample Input	Sample Output	Explanation
`findTriples([3,2,2,4,6,3])`	`12`	Valid choices in terms of (i, j, k) are (1, 0, 3) (1, 0, 4) (1, 3, 4) (1, 5, 3) (1, 5, 4) (2, 0, 3) (2, 0, 4) (2, 3, 4) (2, 5, 3) (2, 5, 4) (0, 3, 4) (5, 3, 4)

6. Statue Race

The wizards and witches of Waverly Palace are preparing for a race between statues. The Palace has **k** statues, each connected through a system of roads[1]. There's only one unique path[2] to get from one statue to another. The only rules are that the participants must follow the system of roads to get from the start statue to end statue (no going off course!). Unfortunately, no one knows where the start or finish statue is for this particular race. Witch Fortune, who will participate in the race, wants to calculate the longest possible race path, measured in the number of roads she must traverse. Given an array of statues that each statue is connected to via a road, help her find this number so she can prepare for the race!

You can assume that there are at least 2 statues.

[1] A road connects two statues.
[2] A path is defined as a traversal of roads from one statue to another.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`getRaceLength(k, arr)`

- **k**: the number of statues in the palace
- **arr**: an array of arrays of integers, where the i^{th} index in **arr** represents ~~palace~~ statue *i* and holds an array of other ~~palaces~~ statues connected to it by a road

Your function should return an **int** that represents the longest race path in terms of the number of roads traversed.

(cont. on next page)

6. Statue Race

Constraints

$2 \leq m.statues \leq 300$

Sample Input	Sample Output	Explanation
`getRaceLength(3, [[1], [0, 2], [1]])`	`2`	A map of Waverly Palace would look like this 0-1-2 And the longest race path is 0->1->2 (or 2->1->0).

`getRaceLength(6, [[1], [0, 2], [1, 3, 5], [2, 4], [3], [2]])`	`4`	A map of Waverly Palace would look like this
		
		0-1-2-3-4
		  |
		  5
		
		And the longest race path is 0->1->2->3->4 (or the reverse like the above test case).

7. Arc of XOR

Gandalf is attempting new spells. Recently, he's been practicing a difficult spell called Arc of XOR, which creates a shield of computer monitors around him. Before he uses it, he is given a sequence of positive integers $a_1, a_2, ..., a_n$. To successfully use the spell, he must find the number of triples of integers, (i, j, k), such that $1 \le i < j \le k \le n$ and $a_i \oplus a_{i+1} \oplus ... \oplus a_{j-1} = a_j \oplus a_{j+1} \oplus ... \oplus a_k$, where $\oplus$ denotes the bitwise XOR operation. Help Gandalf master this spell by solving this problem for him.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`solve(n, a[])`

- **n**: an integer that represents the size of the array a
- **a**: an array containing all the integers

Your function should return an integer that is the number of triples of the above form.

Constraints

$2 \le n \le 10^5$

$1 \le a_i \le 10^6$

Sample Input	Sample Output	Explanation
`xorSub(3, {5, 2, 7})`	`2`	The triples that work are (1, 3, 3) and (1, 2, 3). This is because $5 \oplus 2 = 7$ and $5 = 2 \oplus 7$ (5 is represented as 101 in binary, 2 is represented as 10 in binary, and 7 is represented as 111 in binary; $5 \oplus 2$ = 101 $\oplus$ 010 = 111 = 7; $2 \oplus 7$ = 010 $\oplus$ 111 = 101 = 5.)
`xorSub(4,{0, 0, 0, 0})`	`10`	(1, 2, 2), (2, 3, 3), (3, 4, 4), (1, 2, 3), (1, 3, 3), (2, 3, 4), (2, 4, 4), (1, 2, 4), (1, 3, 4), (1, 4, 4) are all triples that work

8. Potion Brewing

Witch Fortune is trying to brew a potion using a combination of ingredients. First, she needs supplies! On the very top shelf of the storage closet there are **n** ingredients, organized in a line so that a certain ingredient will create very similar effects compared to the ingredient next to it. That means 2 ingredients that are far away from each other cause very different effects. Witch Fortune does not want to select ingredients with similar effects. How many different combinations of ingredients can she select if she wants any ingredient *i* to be at least **k** items away from any other ingredient she chose *j*? Note that she does not necessarily need to use an ingredient!

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`getNumOfPotions(int n, int k)`

- **n:** the number of ingredients on the shelf
- **k:** the minimum distance between any two ingredients

Your function should return the total number of combinations of ingredients to make a unique potion, modulo $10^8 + 7$.

Constraints

$1 \leq k \leq n \leq 20000$

(cont. on next page)

8. Potion Brewing

Sample Input	Sample Output	Explanation
`getNumOfPotions(1, 1)`	2	There is only 1 ingredient so Witch Fortune can either use 1 ingredient or none at all.
`getNumOfPotions(3, 3)`	4	Witch Fortune can either use 0 ingredients, the first one, second, one, or third one. If Witch Fortune uses at least two, the distance between any two will always be less than 3.
`getNumOfPotions(3, 2)`	5	Witch Fortune can use the following combination of ingredients: - None - 0 - 1 - 2 - 0, 2 Note that Witch Fortune cannot use ingredients 0,1 and 1, 2 because the distance between them is 1, which is less than 2.

9. Potion Packing

Oona the sorcerer is packing up for the day and needs to cover all her potion-brewing pots. There are n pots placed on m stoves numbered from 1 to m from left to right, some of which might have a pot on them and others might not. Pot i is placed on stove x_i and not two pots share the same stove.

Oona needs to buy special sheets to cover the pots, each of which can span multiple stoves. A sheet that covers stoves x_i to x_j has length $x_j - x_i + 1$, and it costs c_l to buy a sheet of length l. Larger sheets do not necessarily cost more than smaller sheets.

Help Oona find the minimum cost it takes to purchase a set of sheets so that each pot is covered by at least one sheet. Note that the set of sheets in the optimal solution might overlap to some extent.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`solve(n, m, x[], c[])`

- `n`: an integer that represents the number of pots
- `m`: an integer that represents the number of stoves
- `x`: an array (of size n) where the i-th entry denotes the stove number of the i-th pot
- `c`: an array (of size m+1) where the i-th entry denotes the cost of a sheet of length i.

Your function should return an integer that is the number of triples of the form above.

Constraints

$1 \le n \le 5000$

$1 \le m \le 10^5$

9. Potion Packing

Sample Input	Sample Output	Explanation
`solve(6, 12, {1, 2, 11, 8, 4, 12}, {2, 3, 4, 4, 8, 9, 15, 16, 17, 18, 19, 19})`	9	By purchasing a size 4 sheet, a size 1 sheet, and a size 2 sheet, it is possible to cover all the pots at a cost of 4+2+3=9: SSSSSSS S SSS P P P P P P |---|---|---|---|---|---|---|---|---|---|---|---| 1 2 3 4 5 6 7 8 9 10 11 12 P represents a pot and S represents a part of a sheet.

10. Elixir of Life

You have just stolen the elixir of life from a secret wizard stronghold. Suddenly magic traps have activated around you and there are animated statues that rotate and shoot lasers every second. Since you are a smart and meticulous wizard, you can either wait in your same spot or move in one of the 4 cardinal directions every second (this means you can backtrack!). Given a map of the layout of the stronghold, find out if you can escape, and print the shortest amount of time required if so. (More explanation below!)

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`solve(m)`

- **`m`**: a character matrix that represents the layout of the stronghold. Size = $w * h$
  - . - empty space, can move through
  - S - starting space (your starting location)
  - E - escape point (the end goal)
  - G - animated guard statues that rotate and shoots lasers (that reach the edge of the map) every second. **There can be multiple guards. All guards start facing North** and rotate clockwise 90 degrees every time a second passes. If you are in the direct, straight path of a guard's line of sight after you take a step, you are instantly burned to a crisp.

Your function should return a positive integer indicating the shortest amount of time needed to reach the end if you can, -1 if you cannot reach the end.

Constraints

$w \leq 100,\ h \leq 100$

10. Elixir of Life

Sample Input	Sample Output	Explanation	Explanation (cont.)

`solve(`
`S....`
`.....`
`..G..`
`.....`
`....E)`
	`Escaped in 8 steps.`	A possible shortest path would be:
		
		(X = the laser direction at that moment)
		
		Start:
		`S.X..`
		`..X..`
		`..G..`
		`.....`
		`....E`
		
		`Step 1:`
		`.S...`
		`.....`
		`..GXX`
		`.....`
		`....E`
		
		`Step 2:`
		`..S..`
		`.....`
		`..G..`
		`..X..`
		`..X.E`
		
		`Step 3:`
		`...S.`
		`.....`
		`XXG..`
		`.....`
		`....E`
		
		`Step 4:`
		`..X.S`
		`..X..`
		`..G..`
		`.....`
		`....E`
			(Notice how even though the guard switches from North to East in between steps 4-5 and you are in between the “sweep”, the laser doesn’t kill you; it only shoots once at each NESW direction.)
			
			`Step 5:`
			`.....`
			`....S`
			`..GXX`
			`.....`
			`....E`
			
			`Step 6:`
			`.....`
			`.....`
			`..G.S`
			`..X..`
			`..X.E`
			
			`Step 7:`
			`.....`
			`.....`
			`XXG..`
			`....S`
			`....E`
			
			`Step 8:`
			`..X..`
			`..X..`
			`..G..`
			`.....`
			`....S`

10. Elixir of Life

solve(..S .G. ..E)	Escaped in 3 steps.	The shortest path in this case: Start: .XS .G. ..E Step 1: (decided to wait for 1 sec) ..S .GX ..E	Step 2: ... .GS .XE Step 3: ... XG. ..S

11. TRICK OR TREAT

In order to celebrate Halloween, Jack the Pumpkin King invites n wizards to play a "trick or treat" game. First, he wants every wizard to keep some candies in their left hands and some on their right hands. He also keeps some candies in each of his hands. Next, Jack asks these n wizards to stand in a line, and he will stand at the head of the line.

Now it is time for TRICK OR TREAT!! Let l_i and r_i denote the number of candies in the left and right hands of wizard i respectively. Wizard i is the wizard standing in the i-th place from the front (including Jack; so Jack has l_1 candies in his left hand and r_1 in his right hand). The number of candies that each wizard gets is $\lfloor(\prod_{j=1}^{i-1} l_j)/r_i)\rfloor = \lfloor(l_1 \cdot l_2 \cdot ... \cdot l_{i-1})/(r_i)\rfloor$ where $\lfloor\rfloor$ denotes the floor function.

In other words, the number of candies that each wizard gets is the floor of the product of all the number of candies that wizards before him had in their left hands divided by the number of candies this wizard has in his right hand.

However, Jack does not want one specific wizard to get too many candies. So he wants you to help him rearrange the order of how the wizards line up such that the largest number of candies that any of the wizards get after the rearrangement is as small as possible, i.e. to minimize the maximum number of candies that a wizard can get. Note that Jack the Pumpkin King will always stay at the head of the line.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`trickOrTreat(kL, kR, candies)`

- **`kL`:** the number of candies in king's left hand
- **`kR`**: the number of candies in king's right hand
- **`candies`**: a two-dimensional array, where candiesArr[i][0] gives the number of candies that the ith wizard puts in his left hand, and candiesArr[i][1] gives the number of candies that the ith wizard puts in his right hand.

Your function should return a STRING that corresponds to an integer denoting the minimized maximum number of coins a wizard will get after your rearrangement . (So if the number is 10, your function should return "10".)

11. TRICK OR TREAT

Constraints

1 <= The number of wizards <= 1000;
0 < kL, kR, any entry of candies < 10000

Sample Input	Sample Output	Explanation
`trickOrTreat(1,1,{{2,3},{7,4},{4,6}})`	"2"	If we arrange the wizards in order 1,2,3, then the maximum number of candies that a wizard will get is 2. If we arrange the wizards in order 1,3,2, then the maximum number of candies that a wizard will get is 2. If we arrange the wizards in order 2,1,3, then the maximum number of candies that a wizard will get is 2. If we arrange the wizards in order 2,3,1, then the maximum number of candies that a wizard will get is 9. If we arrange the wizards in order 3,1,2, then the maximum number of candies that a wizard will get is 2. If we arrange the wizards in order 3,2,1, then the maximum number of candies that a wizard will get is 9. Thus, among all cases, the minimum number of largest number of candies that a wizard will get after rearrangement is 2, so the answer is 2.

12. Inversions

The wizard Merlin has a huge collection of crystals with different levels of magical energy. He wants to build a sequence of crystals for his next experiment but he is worried about how unstable his sequences might be. We say that the instability of a sequence is the number of balanced continuous subsequence (intervals). A sequence is balanced if the number of pairs of crystals such that the most powerful crystal appears before the least powerful crystal is at most some value determined by Merlin. We call such pairs inversions, hence the name of the problem.

Implementation Details

Implement the following function. It will be called by the grader once for each test case.

`solve(n, k, list)`

- **`n`:** an integer representing the size of the array
- **`k`:** an integer representing the maximum number inversions desired in each interval
- **`list`:** an integer array of size **n**. This array contains every integer from **1** to **n** exactly once. Crystals with a larger number are more powerful than crystals with a smaller number.

Your function should return an integer representing the instability of the given sequence. Since this value can be very large you should use **long** instead of **int** when working in Java.

Constraints

$1 \leq n \leq 10^6$

$1 \leq k \leq n(n-1)/2$

(cont. on next page)

12. Inversions

Sample Input	Sample Output	Explanation
4 0 3 1 4 2	5	Subsequences [3], [1], [4], [2] and [1 4] don't have inversions but any other subsequences will have at least one.
4 3 4 3 2 1	9	There are 10 possible subsequences and only one has more than 3 inversions, namely the whole sequence.