Contest Problems
Philadelphia Classic, Spring 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.

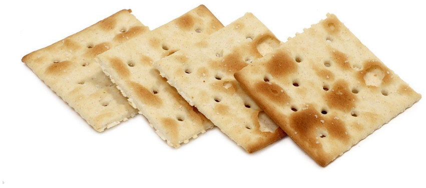
There is no penalty for incorrect submissions. You will receive 1 point per problem solved. A team’s number of incorrect submissions will be used only as a tiebreaker.

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. Eating Contest



Matt has spent the past two months preparing for his saltine cracker eating contest against Joey. On the day of the contest, Matt calls his friend Bob to keep track of how many crackers they each eat. Matt and Joey agree to compete for multiple rounds, each taking 15 minutes, to make sure there is a clear winner for who can eat more saltine crackers. Unfortunately, Bob can only write scores from left to right, with Matt's score for the first round first and Joey's score for the first round second, followed by Matt's score for the second round and Joey's score for the second round, and so on. Help Matt and Joey identify the winner of the entire contest.

Implementation Details

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

`winner(scores)`

- **`scores`**: an integer array that represents the scores

Your function should return a lowercase string of the name of the winner ("matt" or "joey"), or "tie" if there is a tie.

Constraints

$2 \leq |scores| \leq 1000$, and is even

Sample Input	Sample Output	Explanation
`winner([12, 15, 13, 10, 7, 3, 3, 7])`	`tie`	Matt's scores 12, 13, 7, and 3 add up to 35, and Joey's scores 15, 10, 3, and 7 also add up to 35
`winner([232, 10, 1, 484])`	`joey`	Joey has a total score of 494, greater than Matt's total score of 233

2. Morning Buns

A secret admirer has sent you a batch of morning buns for Valentine's Day! Just before you eat one, though, you realize each of the buns have letters on them. They appear to be messages that can be deciphered by reading the letters from the outside to the inside, clockwise. Given a morning bun as a matrix input, can you "unravel" it and return the whole message?

To be more specific, you will be given a square matrix of characters with a message that starts at the top left and moves inwards in a clockwise fashion. There will be no spaces in between characters of the message, but there might be asterisks that must be converted to spaces. Any spaces at the end of the message that were not originally asterisks should be removed. (This means that any spaces that are at the end of the message that were originally asterisks should be returned.) Any characters including letters and punctuation, asterisks and spaces will be in the matrix. Capitalization should be preserved.

Implementation Details

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

`message_reveal(matrix)`

- **`matrix:`** a square 2D array of characters

Your function should return the unraveled string.

Constraints

$1 \leq s \leq 500$, where s is the side length of the 2D array

(cont. on next page)

2. Morning Buns

Sample Input	Sample Output	Explanation
message_reveal([[‘I’, ‘*’, ‘l’, ‘o’, ‘v’], [‘o’, ‘u’, ‘*’, ‘s’, ‘e’], [‘y’, ‘ ’, ‘ ‘, ‘m’, ‘*’], [‘*’, ‘e’, ‘l’, ‘i’, ‘t’], [‘e’, ‘e’, ‘s’, ‘*’, ‘o’]])	I love to see you smile	The array looks like the following. I*lov ou*se y m* *elit ees*o Starting from the top left and working inwards clockwise, you get I*love*to*see*you*smile-- (where - is a space, dash is just in this explanation for clarity!). Removing the original spaces at the end and replacing the asterisks with spaces and gets you to the right output.
message_reveal([[‘I’, ‘\’’, ‘m’, ‘*’, ‘s’, ‘o’, ‘*’], [‘*’, ‘t’, ‘o’, ‘*’, ‘m’, ‘e’, ‘h’], [‘e’, ‘*’, ‘l’, ‘i’, ‘k’, ‘e’, ‘a’], [‘l’, ‘e’, ‘u’, ‘.’, ‘e’, ‘t’, ‘p’], [‘b’, ‘n’, ‘o’, ‘y’, ‘*’, ‘*’, ‘p’], [‘a’, ‘o’, ‘e’, ‘m’, ‘o’, ‘s’, ‘y’], [‘*’, ‘s’, ‘a’, ‘w’, ‘*’, ‘I’, ‘*’]])	I'm so happy I was able to meet someone like you.	The array looks like the following. I'm*so* *to*meh e*likea leu.etp bnoy**p aoemosy *saw*I* Starting from the top left and working inwards clockwise, you get I’m*so*happy*I*was*able*to*meet*someone*like*you. (no spaces in the middle here!). Replacing the asterisks with spaces gets you to the right output.

3. Balanced Diet

Pikachu is an organized eater. He keeps track of when he starts eating a certain food, and when he finishes it. Lately, he has been on a “balanced foods” diet, where specific foods must be eaten in pairs. There are three kinds of balanced food pairs: `{} [] ()`. For Pikachu’s diet, there are two conditions:

1. It contains no unmatched foods.
2. The subset of balanced foods within another pair of balanced foods is also a “balanced food”.

Implementation Details

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

`isBalanced(input)`

- **`input:`** a String containing `‘{‘, ‘(‘, ‘[‘, ‘]’, ‘)’, ‘}’` (no quotation marks) characters less than 10,000 character in length

Your function should return “True” or “False” indicating whether or not the “food” is balanced.

Sample Input	Sample Output	Explanation
`isBalanced(“{[()]}”)`	`True`	Each bracket is properly balanced.
`isBalanced(“{)”)`	`False`	The “{“ isn’t closed, the “)” isn’t opened.

4. Recently Vegan

One day, Bob is very hungry and decides to visit a restaurant. Each item on the menu is labeled with the number of calories, and ordered from least amount of calories to most amount of calories. It just so happens that the calories for items on the menu are consecutive positive integers starting from 1. Just the day before, however, he chose to become vegan. Fortunately, Bob finds a strange guide to which items at the restaurant are vegan. If the number corresponding to a food item is divisible by a number on the guide, it is a vegan option. Given a maximum menu item number **n** corresponding to Bob's calorie limit, find the number of menu items Bob is allowed to eat.

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 max menu item number that is within Bob's calorie limit
- **A**: an array of sorted integers (**This is an array of strings**), the "guide" to finding which items are vegan

Your function should return the number of numbers from 1 to **n** that are divisible by at least one element in **A**.

Constraints

$1 \le n \le 2500$, $1 \le |A| \le 1000$

(cont. on next page)

4. Recently Vegan

Sample Input	Sample Output	Explanation
`solve(5, [3, 6, 7, 12])`	`1`	The food menu items are: 1, 2, 3, 4, 5. 3 is the only number less than or equal to 5 that is divisible by a number in the array (in this case, 3).
`solve(10, [2, 3, 4, 5, 6])`	`8`	The food menu items are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. 2, 3, 4, 5, 6, 8, 9, and 10 are numbers less than or equal to 10 divisible by at least one element in array. 2 - divisible by 2 3 - divisible by 3 4 - divisible by 4, or 2 5 - divisible by 5 6 - divisible by 6, 2, or 3 8 - divisible by 2, or 4 9 - divisible by 3 10 - divisible by 2, or 5

5. Waitress' Dilemma

Zeach was recently employed as a waitress at the Rordon Gamsay's Great Restaurant. Rordon wanted a new and modern design for his restaurant like no other before and decided to employ Wowser to be his architect. Wowser came up with this brilliant idea of making all the tables float in midair. According to Wowser's drawings, one can describe each of the n^2 tables with a tuple (x, y) where $1 \le x, y \le n$, and $n \ge 6$ is an even number. However, as the infamous cheeky architect, Wowser only built bridges from any two tables $(x_i, y_i),\ (x_j, y_j)$ if $|x_j - x_i| = 2$ and $|y_j - y_i| = 1$, or if $|x_j - x_i| = 1$ and $|y_j - y_i| = 2$. Zeach was sent up to table (x_s, y_s) by her good friend Pelda, but Pelda has some other important business to attend to. Help Zeach determine the minimum number of bridges she must cross to reach table (x_d, y_d) before Rordon starts yelling!

Implementation Details

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

```
minBridges(n, xs, ys, xd, yd)
```

- **n**: an integer such that there are n^2 tables
- **xs**: an integer that represents the starting x-coordinate
- **ys**: an integer that represents the starting y-coordinate
- **xd**: an integer that represents the ending x-coordinate
- **yd**: an integer that represents the ending y-coordinate

Your function should return an integer that is the minimum number of bridges Zeach will have to cross to reach her destination.

Constraints

$6 \le n \le 300$ and is even

(cont. on next page)

5. Waitress’ Dilemma

Sample Input	Sample Output	Explanation
`minBridges(6, 1, 5, 1, 3)`	2	Zeach only has to cross 2 bridges by taking the following path. `(1,5) -> (3, 4) -> (1, 3)` $\lvert x_2 - x_1 \rvert = \lvert 3 - 1 \rvert = 2$ $\lvert y_2 - y_1 \rvert = \lvert 4 - 5 \rvert = 1$ and $\lvert x_3 - x_2 \rvert = \lvert 1 - 3 \rvert = 2$ $\lvert y_3 - y_2 \rvert = \lvert 3 - 4 \rvert = 1$

6. Goldilocks and the Triplet Bears

Goldilocks has befriended the three bears. Now the three bears are hungry! However, Papa Bear must eat exactly 3 times as much as Mama Bear, and Mama Bear must eat exactly 3 times as much as Baby Bear. Additionally, Papa bear must eat after Mama and Baby bear (order between Mama and Baby doesn't matter). Given an array representing the amount of food available in a line, figure out how many ways there are such that:

1. Papa Bear eats at index *a* and 3 times as much as Mama Bear
2. Mama Bear eats at index *b* and 3 times as much as Baby Bear at index *c*
3. Mama Bear and Baby Bear eat before Papa Bear (order of Mama and Baby doesn't matter). (*b, c* < *a*)

Implementation Details

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

`tripleBears(arr)`

- **`arr:`** an integer array of positive integers that will have at least one element.

Your function should return an integer that represents the number of ways the three bears can eat such that the conditions are true.

Constraints

$|arr| \leq 10000$

(cont. on next page)

6. Goldilocks and the Triplet Bears

Sample Input	Sample Output	Explanation
`tripleBears([1, 3, 9])`	`1`	There is one way to satisfy the conditions. Papa Bear will eat at index a = 2, Mama Bear will eat at index b = 1, and Baby Bear will eat at index c = 0.
`tripleBears([1, 2, 3])`	`0`	There is no way to satisfy the conditions.

7. The Cupcake Club

On a whim, you have joined the PCC (Penn Cupcake Club)! PCC is doing a fundraiser to raise money for their intense, cutting-edge research projects (including but not limited to creating the perfect icing and researching the optimal temperature for baking cupcakes). They have decided to take orders for messages to be displayed by one letter per cupcake (look at photo for reference). As a member of the club you do your part to help make these orders. However, just as you've completed transcribing a message onto cupcakes you realize you were reading the completed messages list instead of the to-do list! Worried that you won't have enough time to redo the order completely, can you calculate how many changes you will need to make to change the message you made into the message you need to make?

Your goal is to change message$_1$ into message$_2$. You can perform any of these 3 operations on message$_1$:

1. Add a new cupcake to the message (Insert a character)
2. Remove a cupcake from the message (Delete a character)
3. Remove a cupcake from the message and add a new cupcake in the same spot in the message (Replace a character)

Implementation Details

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

```
solve(message1,message2)
```

- **`message1`**: a string that is the incorrect order you fulfilled
- **`message2`**: a string that is the correct order you should fulfill by transforming message1 with the operations listed above

Your function should return the minimum number of operations required to turn message1 into message2.

Constraints

$1 \le s \le 1000$, where s is the length of any string (message1 or message2)

7. The Cupcake Club

Sample Input	Sample Output	Explanation
`solve(“horse”, “ros”)`	3	Replace h with r (horse -> rorse) Remove r (rorse -> rose) Remove e (rose -> ros)
`solve(“intention”, “execution”)`	5	remove 't' (intention -> inention) replace 'i' with 'e' (inention -> enention) replace 'n' with 'x' (enention -> exention) replace 'n' with 'c' (exention -> exection) insert 'u' (exection -> execution)

8. Picnic Day

Your good friend Braydon owns a collection of small islands that are connected by bridges. The bridges are arranged such that there is only one set of bridges a person can take to travel between any two particular islands, and it is possible to reach any other island from any specific starting island. You can travel both ways on trees. He is currently planning a picnic, where each island will serve as a "food station" for a particular dish. However, to ensure the event has variety, he wants to set up the food stations so that any island is always at least distance 3 away from any other island with the same food dish. What is the minimum number of different types of dishes he must cook to ensure that his picnic is exciting?

Implementation Details

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

`minDishes(n, bridges)`

- **`n:`** the number of islands
- **`bridges:`** a list of size-2 lists of numbers which represent the endpoints for each bridge.

Your function should return the minimum number of dishes Braydon needs for his party.

Constraints

$3 \leq n \leq 20000$

Output Format

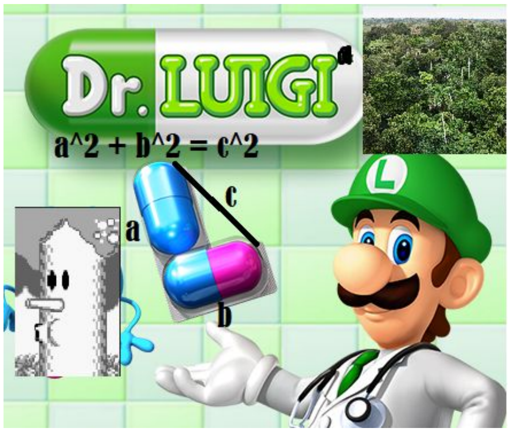
Output an integer representing the minimum number of dishes Braydon must cook.

(cont. on next page)

8. Picnic Day

Sample Input	Sample Output	Explanation
`minDishes(5, [[0,1],[1,2],[2,3],[3,4]])`	3	The islands are all connected in a linear fashion. Island 4 can have the same dish as Island 1, and Island 2 can have the same dish as Island 5.
`minDishes(4, [[0,1],[0,2],[0,3]])`	4	All islands are within distance 2 of each other, so all 4 islands need different dishes.

9. Exotic Orchard



Dr. Luigi owns an orchard consisting of **N** exotic tropical trees. In order to keep track of where all of his trees are, he has a single stretchy elastic band that is wrapped around all of his trees, forming a convex (all interior angles less than 180 degrees) polygon with a perimeter that encloses all of his trees. One day, he decides to plant a new tree at a certain coordinate; however, he wants to make sure that the new tree is on or within the perimeter that is marked by his elastic band. Write a function that, determines if his new tree will be within the perimeter.

Implementation Details

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

`in_perimeter(points, new_point)`

- **`points`:** list of **N** Points (int x, int y) in Java or size-2 tuples in Python representing the existing trees.
- **new_point**: Custom Point class (int x, int y) in Java or one size-2 tuple in Python representing the (x, y) location of a potential new tree.

Your function should return “True” if the new tree is inside the perimeter and “False” if it is not.

Constraints

$3 \leq N \leq 10000$

(cont. on next page)

9. Exotic Orchard

Sample Input	Sample Output	Explanation

`in_perimeter(`
`    [(1,1),(-1,1),(1,-1), (-1,-1)],`
`    (0,0)`
`)`
	`True`	The coordinate (0, 0) is within the smallest convex polygon that encloses all of the given vertices.

`in_perimeter(`
`    [(1,1),(-1,1),(1,-1), (-1,-1)],`
`    (2,2)`
`)`
	`False`	The coordinate (2, 2) is not within the smallest convex polygon that encloses all of the given vertices.

10. Hell's Immigration

Rordon Gamsay, a world renowned Michelin-starred chef, realizes that he is very late for the debut of his critically acclaimed show, "Hell's Kitchen", in his newly bought town, Gamsaytown.

Gamsaytown has **N** cities (numbered 1 to **N**) and **M** bidirectional roads. Road **i** takes connects two cities u_i and v_i and takes w_i minutes to traverse. It is guaranteed that there exists a combination of roads that connect any pair of cities.

Rordon is at city **s** and wishes to go to the studio of "Hell's Kitchen" which is located at city **d**. Due to security issues Gamsaytown has had in the past (in particular, people stealing cookbooks from the city), one has to go through a customs check before leaving a city.

Each city has exactly two officers who works on shift. In particular, for a city **c**, the shift cycle of the officers can be described as an ordered pair (l_c , h_c), where the first officer works for l_c minutes, before the second officer takes over and works for h_c minutes, and the cycle continues. However, knowing that Rordon will be insulting the poor contestants in "Hell's Kitchen", the second officer of each city made a deal to stop Rordon from leaving the city if they are on shift. Note that the first officer starts the shift at minute 0, the point at which Rordon leaves the first city. Assume that the shift change is instantaneous. Note that it is always possible to go from city **s** to city **d**.

Rordon, being the hot-tempered person he is, is very angry when he found out what had happened. Help him determine the shortest time it would take for him to reach the studio before he throws a piece of raw salmon!

Implementation Details

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

```
helpRordan(n, m, shift, edges, s, d)
```

- **n**: an integer that represents the number of cities. The cities are labeled 1 to n inclusive.
- **m**: an integer that represents the number of roads

- **`shift`**: a list of pairs (tuple in Python, a custom Shift object in Java) such that the i^{th} pair contains l_i, h_i where the former is the length of the first officer's shift and the latter is the length of the second officer's shift for city i
- **`edges`**: a list of tuples (tuple in Python, a custom Edge object in Java) such that the i^{th} tuple contains u_i, v_i, and w_i, where the triplet represents an edge from city u_i to city v_i with weight w_i.
- **`s`**: an integer that represents the starting city
- **`d`**: an integer that represents the destination city

Your function should return an integer that is the shortest amount of time Rordon can reach his studio.

Constraints

$n \le 500,\ w_i \le 30000,\ l_c,\ h_c \le 100$

Sample Input	Sample Output	Explanation
`helpRordan(4, 4, [(1,1), (1,3), (2,2), (2, 10)], [(1, 2, 1), (1, 3, 1), (2, 4, 3), (3, 4, 4)], 1, 4)`	5	There are two possible routes to travel from city 1 to city 4, namely through 2 and 3. Travelling through city 2 takes 1 + 3 + 3 = 7 minutes as the time on road is 1 + 3 and Rordan has to wait at city 2 for 3 minutes until the first officer comes back. Travelling through city 3 takes 1 + 4 = 5 minutes as the time on road is 1+4 and the first guard is still on shift at minute 1 when Rordan reaches city 3. The shortest time is hence 5 minutes.

11. Stolen Cookies

Your roommate is a terrible person. He stole all your cookies and will only give them back if you help him with his homework.
He has to evaluate a complicated numerical expression. This expression is made of many smaller expressions in a recursive manner.

First, we have boolean expressions. They are of the form “T”, “F”, “NUM1<NUM2”, “NUM1=NUM2” or “NUM1>NUM2” where “NUM1” and “NUM2” are numerical expressions. They will evaluate to a boolean value of true (“T”) or false (“F”).

We also have numerical expressions. They can be either a positive number “1”, “2019” or “1337” for example, an addition of the form “NUM1+NUM2” where “NUM1” and “NUM2” are numerical expressions and it evaluates to the sum of the evaluations of “NUM1” and “NUM2”, or a conditional statement. A conditional statement is of the form “(BOOL?NUM1:NUM2)” where “BOOL” is a boolean expression and “NUM1” and “NUM2” are numerical expressions and it evaluates to the evaluation of “NUM1” if “BOOL” evaluates to true and to the evaluation of “NUM2” otherwise.

Numerical evaluations have priority over boolean evaluations.

Implementation Details

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

```
solve(str)
```

- **`str`**: A string of characters in the list `{“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “T”, “F”, “<”, “+”, “>”, “=”, “(”, ”)”, “:”, “?”}`.

Your function should return the correct evaluation of the numerical expression given by the string.

Constraints

$1 \leq |str| \leq 100000$

(cont. on next page)

11. Stolen Cookies

Sample Input	Sample Output	Explanation
`solve(“(1+2>2+2?4:5)”)`	5	“1+2” and “2+2” evaluate to “3” and “4” respectively so we have “(3>4?4:5)” where “3>4” evaluates to “F” and finally we have “(F?4:5)” which evaluates to “5”.
`solve(“(F?497:1)+5+(F?67:12)+(F?167:48+143+36+453)”)`	698	“`(F?497:1)`” evaluates to “1”, “`(F?67:12)`” evaluates to “12” and “48+143+36+453” evaluates to “680”. Then, `(F?167:680) evaluates to 680 and “1+5+12+680” evaluates to 698.`

12. Gather the Bananas

Konkey Dong wants to gather the most bananas in the most efficient way possible. The jungle is littered with bananas and Konkey Dong recorded the coordinates of all of them. He wants you to find the circle with the smallest radius such that he can see all of the bananas at once. Your job is to find the the smallest circle that encloses all the points. If there is only one point, then just output that point with radius 0.

Implementation Details

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

`makeCircle(points)`

- **`points:`** a list of
  - Java - Point objects created by a Point class (double x, double y)
  - Python - 2-tuples (float x, float y)

Java - your function should return size-3 array of doubles in the order of (x, y, r)
Python - your function should return a size-3 tuple (x, y, r) of floats
(x, y, r) corresponds to (x-coordinate of center, y-coordinate of center, radius)

Constraints

$1 \leq |points| < 6000$

(cont. on next page)

12. Gather the Bananas

Sample Input	Sample Output	Explanation
`makeCircle([(2, 3)])`	`(2.0, 3.0, 0.0)`	The coordinate is just a single point, so the smallest circle is centered at that point with radius 0.
`makeCircle([(-1, 0), (1, 0)])`	`(0.0, 0.0, 1.0)`	The coordinates are (-1, 0) and (1, 0). The smallest circle is centered at the origin with radius 1.
`makeCircle([(86.57181715032996, 47.88177329585104), (55.964165942856056, 69.78412798728033), (96.53990265903957, 69.44840386267505)])`	`(76.22, 65.753, 20.653)`	Here is an example featuring floats. Three non-collinear points define a circle since they also define a triangle. The center is the triangle's circumcenter and the radius is its distance to the vertices.