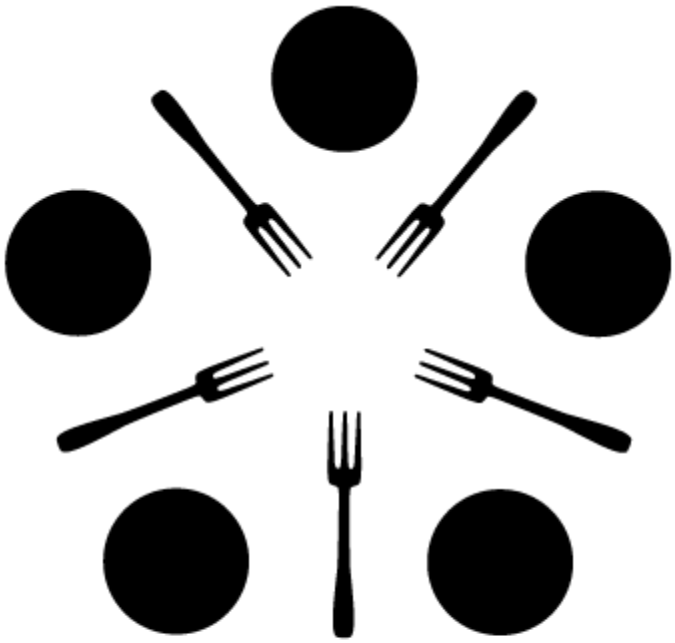
Contest Problems
Philadelphia Classic, Fall 2018
Hosted by the Dining Philosophers
University of Pennsylvania



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.

when your code doesn’t compile with the sample test cases but you submit anyways and get the problem wrong

←-- A friendly reminder you should not do this.

Disclaimer: We are not affiliated with Nintendo or Pokemon in any way. We do not own Pokemon or the images used in this packet. All rights belong to the original owners/creators.

1. Nicknames

Ash is terrible at nicknaming his Pokemon. He has given you a proposed nickname and your job is to tell him why his nickname is terrible by pointing out the longest consecutive appearance of any character in the name. If there are multiple sequences of the same length, return the first one. You can assume the name only contains alphanumeric characters (a-z, A-Z, 0-9). Note, lowercase characters are **NOT** the same as uppercase characters.

Input Format

A string of alphanumeric characters. The string will be at least 1 character long and at most 20,000 characters long.

Output Format

The longest consecutive sequence of characters in that string.

Sample Input	Sample Output	Explanation
`aaabbbaaaabbb`	`aaaa`	“a” appears four times in a row, which is the maximum consecutive substring.
`bbbbbaaaaab`	`bbbbb`	While both “b” and “a” appear five times in a row, the “b” sequence appears first.

2. Proximity

There are many possible places within the grass that you can travel. However, it would be helpful to know the locations of any nearby wild Pokemon at each of your potential travel locations. Given a list of possible travel locations, we will try to identify the two closest Pokemon from a known list of Pokemon locations.

Input Format

An integer array L consisting of one or more integers, representing the known Pokemon locations, and another integer array M consisting of one or more integers, representing potential travel locations. L and M will be at most 50 integers each.

Output Format

An integer array that is twice as long as the number of elements in M that contains the two closest Pokemon locations for each travel location in M. The two closest integers should appear with the closer number first. If two integers in L are equally close to an integer in M, the smaller number should be chosen first. Pair order should correspond to the order integers appear in M.

Sample Input	Sample Output	Explanation
`[1, 2, 4, 5]` `[2, 7, 3]`	`[2, 1, 5, 4, 2, 4]`	Returns the two closest integers for each integer in the second list. The first location is 2, so the closest pokemon to that location are 2 and 1 in order of closeness. The second location is 7, so the closest pokemon to that location are 5 and 4 in order of closeness. The third location is 3, so the closest pokemon to that location are 2 and 4 in order of closeness (because they are equally close in this case 2 is put first because it is the smaller number).
`[1]` `[2, 3, 4]`	`[1, 1, 1, 1, 1, 1]`	If the first list only has one integer, it is the only option.

3. Beat That Clefairy

Archer, the new leader of Team Rocket, wants to steal millions of rare candies to beef up his Pokemon. He must beat five Clefairies in a set number of moves to battle Lady Celeste, whose Pokemon holds rare candies. He is down to the last Clefairy and has given you a list of moves he can use, as well as the Clefairy's remaining health. He has three moves left to use and wants to know how many combinations of moves he can use to reduce the Clefairy's health down to 0.

Input Format

The Clefairy's starting health on the first line. The second line is the integer list of damage calculations separated by a comma and space. Note that in the input boxes below they appear to be on multiple lines, but they are all on one line in the test case.

Output Format

An integer for the total number of ways he can drop the Clefairy's health down to exactly 0.

Sample Input	Sample Output	Explanation
3 1, 2, 3	0	The only combination of damage is 1+2+3 which is not equal to 3.
100 50, 92, 88, 6, 25, 9, 100, 55, 0, 8, -25	2	The combinations are 92+0+8 and 25+100-25. Note that the order doesn't matter (the second choice goes over 100 before going back down) since we could easily rearrange it to be 25-25+100 instead.

4. Career Pivot

After a series of brutal losses to various gym leaders, Ash and Gary have unfortunately decided to pivot from their original career and study computer science. In class, they learned about binary numbers and converting between decimal and binary. However, Team Rocket has still not given up on Ash's Pikachu, and managed to steal and lock Pikachu up in a cage. As usual, Gary comes to the rescue with his Umbreon and finds the cage in a forest. On the cage, there is a list of integers and a mysterious operation called PKM with the following definition:

PKM is an operation between two numbers in binary form, A and B, where the ith digit of A PKM B is 1 if and only if exactly one of the ith digits of A or B is 1. For example:

001 PKM 011 = 010

In order to unlock the cage, the password requires the minimum result of using the PKM operation on any two integers in the list in decimal form. Save Pikachu!

Input Format
Array of positive integers that contains at least two elements with bounds of: 0 <= x <= 10^6

Output Format
Return the minimum value (**NOT** the pair that produces the minimum)

Sample Input	Sample Output	Explanation
`[5, 3, 4, 2, 1]`	`1`	2 = 10 3 = 11 10 PKM 11 = 01 = 1 Choosing 2 and 3, the operator will return 0 for the tens place and 1 for the ones place, resulting in 1 in decimal form which is the minimum possible value.
`[10, 13]`	`7`	10 = 1010 13 = 1101 1010 PKM 1101 = 0111 = 7 The only two possible values are 10 and 13, in which the operator will return 7 in decimal form.

5. Portals

Dawn is stuck in Team Galactic's headquarters and needs to get out quickly. There might be portals around that can help her escape. Dawn can only move in the 4 cardinal directions, North, East, South, or West, and each step takes 1 second. Travelling through a portal, though, is instantaneous.

`.` = Empty space, can travel through
`#` = Wall, cannot travel through
`$` = Starting location
`&` = Ending location
`A, B, C, D, ...` = Portals

There will be at most 26 pairs of portals (capital letters of the alphabet). You can travel into either portal and get out of the matching paired portal. The portals will always be paired with another with the same capital letter. There can be as few as 0 portals. The portals are named in increasing alphabetical order: if there are 3 portals for example they will always be called `A, B,` and `C` pairs, not arbitrary letters. However, the location they appear in the maze is not necessarily in order.

Note that there will always be a start, end, and a possible escape route, so there will always be a distance to print. Print out the fastest time that Dawn can get through the maze and escape.

Input Format
An integer `n` which is the number of test cases
For each test case:
An integer `r` and `c` which are number of rows and columns respectively.
4 <= `r` <= 100, 4 <= `c` <= 100
Next `r` lines are the map of Team Galactic's headquarters

Output Format
A single integer which is the fastest time Dawn can get through the maze and escape.

Sample Input	Sample Output	Explanation

2
12 10
##########
#$.......#
#........#
#.....A..#
#........#
#..B.....#
#........#
#........#
#..A..B..#
#........#
#.......&#
##########
12 10
  ######
  #....#
  #....#
####..A..#
#........#
#..B.....#
#......#.#
#......#.#
#..A..B#.#
#...####.#
#.$.# #&#
##### ### 	
	10
	12
		Dawn can take 6 seconds to get to the Portal B that is closest to her, then from the other Portal B she takes 4 more seconds to reach the exit.
		
		Dawn can take 3 seconds to get to the Portal A that is closest to her, then from the other Portal A she takes 9 more seconds to reach the exit.

6. Gotta Catch ‘Em All

Team Rocket has stolen Ash’s Pikachu (for the millionth time), and demands a ransom. He must catch a specific Pokemon to trade for his beloved Pikachu. Team Rocket has given him a list of Pokemon numbers as well as the number that they want. Being the helpful villains they are, Team Rocket decided to sort the rows and columns. This means that for every number i, every other number on the same row as i and to its left is less than i, and every number to its right is greater than i. Moreover, every other number on the same column as i and above it is less than i, and every number below it is greater than i. Help Ash find that ransom Pokemon.

Input Format

The integer you are looking for on the first line. The number of rows (N) on the second line. N more lines each serving as a row of the 2-D array (ex. 1, 2, 3, 4, 5 are the five numbers on the first row). The length of the arrays will be at least one and less than or equal to 250.

Output Format

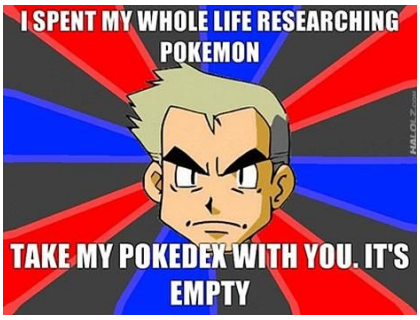
The index of the integer (separated by comma, no spaces). If not in the array, output “Pikachu is lost”

Sample Input	Sample Output	Explanation
15 3 1, 2, 3, 4, 5 6, 7, 8, 9, 10 11, 12, 13, 14, 16	Pikachu is lost	15 is not part of the 2-D array.
44 6 10, 20, 30, 40, 50 11, 21, 31, 41, 51 12, 22, 32, 42, 52 13, 23, 33, 43, 53 14, 24, 34, 44, 54 15, 25, 35, 45, 55	4, 3	44 is in the fifth row and fourth column. Note that indexes start at 0.

7. Professor Oak's Palindromes

A little known fact about Professor Oak is that he LOVES palindromes. Therefore, it's obvious that he loves Pokemon with palindrome names (random fun fact: there are only 4 Pokemon with palindromic names: Eevee, Girafarig, Ho-Oh, and Alomomola). Recently Oak has discovered a completely new region with unknown Pokemon species. He plans to name each new species with a palindromic name and attempts to write a program that will generate palindromes to be potential names. However, Professor Oak doesn't write code very well (his PhD is in Pokemon Studies, not Computer Science), so his program actually just prints a string of random characters and random length. Given one of Professor Oak's random strings, can you look through it and return the longest palindromic substring? The longest palindromic substring could be as short as 1 character or as long as the entire string. If there are multiple longest palindromic substrings with the same length then return the one that comes first in the string (in other words closer to the beginning of the string).

The random string will only contain lowercase letters.



Input Format
An integer `n` which is the number of test cases.
For each test case:
A string `s` which is the random string that Professor Oak's program generated. The string will be no longer than 1000 characters and at least 1 character long.

Output Format
A string which is the longest palindromic substring.

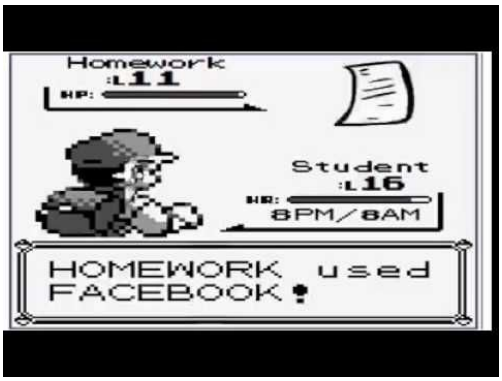
Sample Input	Sample Output	Explanation

`3`
`fwsanavanaklcdks`
`abc`
`racecar`
	`anavana`
	`a`
	`racecar`
		`anavana` is the longest palindromic substring
		
		`a` is the longest palindromic substring that is closest to the beginning
		
		`racecar` is a palindrome by itself, so the longest palindromic substring is the whole string

8. Homefun, not Homework

Ash is terrible at math (which is probably why he eventually dropped out of school to become a Pokemon trainer). He promises to give you 100 PokeDollars if you do his homework for him (note: we won't actually give you 100 PokeDollars to solve this problem). Since you are a very busy person (but you want the money!), you decide to write a program to solve the problems.

Equations are given in the format A / B = k, where A and B are variables represented as strings, and k is a real number (floating point number). Given some queries, return the answers. If the answer does not exist, return -1.00.



Example:
Given a / b = 2.0, b / c = 3.0.
queries are: a / c = ?, b / a = ?, a / e = ?, a / a = ?, x / x = ? .
return [6.0, 0.5, -1.0, 1.0, -1.0].

According to the example above, these are the arrays you will be given in the stub method:
equations = [["a", "b"], ["b", "c"]],
values = [2.0, 3.0],
queries = [["a", "c"], ["b", "a"], ["a", "e"], ["a", "a"], ["x", "x"]].

The input is always valid. You may assume that evaluating the queries will result in no division by zero and there is no contradiction. You may assume that 0 will not be an answer to any of the equations. NOTE: If a query variable was not listed in a prior given equation, you cannot evaluate it at all. The answer should be -1.00.

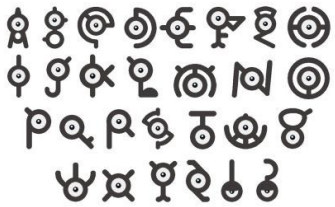
The answers will automatically be rounded to the second decimal point and printed on separate lines in the output based on the array of answers you send in. There should be NO rounding during calculations.

Input Format
An integer `n` which is the number of given equations
For each given equation:
A string `A`, a string `B`, and a double `k` that should be read as A / B = k
1 <= `k` <= 100

Output Format
An answer for each query on their own line, in order of given queries.

Sample Input	Sample Output	Explanation
2 a b 2.0 b c 3.0 5 a c b a a e a a x x	6.00 0.50 -1.00 1.00 -1.00	a / c = (a / b) * (b / c) = 2.0 * 3.0 = 6.0 b / a = (a / b) ^ (-1) = (2.0) ^ (-1) = 0.5 a / e = -1.0, since e does not exist in any given equations a / a = 1.0, because any non zero number over itself is 1 x / x = -1.0, since x does not exist in any given equations. This should not evaluate to 1.0.

9. Strange Unowns



Lucas is trying to make a team of *k* unowns from a list of *n* total unowns. He is very particular about the type of team he wants, and specifically wants unowns with attack levels that are very close to each other. In addition, the unowns are already in a particular order in the list, and Lucas wants his team of *k* unowns to be next to each other in the list. Lucas would like to see all of the options he has, so that he can make a better decision. Given a list of n attack levels, with each attack level corresponding to a unown, show the difference between the highest and lowest attack level of each team that Lucas can make. Help Lucas make his dream team!

Input Format

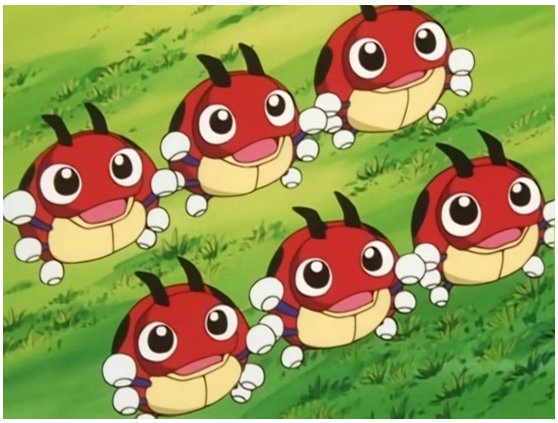
A non-negative integer, *k*, indicating the size of the team that Lucas wants to make, followed by a list of integers of size *n* indicating the unown attack levels. *k* is greater than 2, and *n* is greater than or equal to *k*. 2 <= *k* <= *n* <= 10,000,000

Output Format

A list of integers indicating the difference in attack level for each possible team of size *k*.

Sample Input	Sample Output	Explanation
`2 1 1`	`0`	There is only one possible team in this case, with unowns with attack level 1 and 1. The difference between these attack levels is 0.
`5 1 2 3 4 5 6`	`4 4`	There are two possible teams in this case, and the difference within each team is 4. 5 - 1 = 4, 6 - 2 = 4

10. Rectangles



A pack of Ledyba requires your assistance. They are looking for a new grass field to live in, but since the fields can be very large and have many unusable areas, they need your help. Ledyba love berries but hate rocky areas, so they want to know the total area with berries that is not rocky.
Your input will contain a series of rectangles, each rectangle will be described by the coordinates of its lower left corner and upper right corner. First you will have the rectangle which represents the whole field. Then there will be two lists, one with the rectangles corresponding to the rocky areas and another corresponding to the areas without berries. Your task is to find the total area inside the field which has berries and isn't rocky (you just need to do it one field at a time, the Ledyba will decide which one to move in based on other factors also).
Luckily you know that there is no intersection between any two rocky areas in the lists nor between any berryless areas. However, it may happen that some of them are outside the grass field, so pay attention.

Input Format
An integer representing the number of test cases.
One array of size 4 containing the coordinates of the whole field rectangle in the following format: (x_0, y_0, x_1, y_1) where (x_0, y_0) are the cartesian coordinates of the lower left corner of the rectangle and (x_1, y_1) the cartesian coordinates of the upper right corner of the rectangle.
One list of arrays of size 4 containing the coordinates of rectangles without berries in the same format above.
One list of arrays of size 4 containing the coordinates of rectangles with rocky area in the same format above.
Notes: There will be 10 test cases of increasing number of rectangles in the lists. The last case will have 10^4 rectangles in each list. You may assume that the area of the rectangle is less than or equal to 10^8 in all test cases.

Output Format
An integer indicating the total area.

Sample Input	Sample Output	Explanation
2 0 0 4 4 1 1 0 0 2 3 3 2 4 4 1 1 6 6 1 1 1 0 5 5 2 4 7 7	8 4	For the second test, the field has area 16 but 6 of it has no berries and 2 of it is rocky. No area is both rocky and berryless. For the second test, the field has area 25 but 16 of it has no berries, 8 of it is rocky and 3 of it is both rocky and berryless. Thus there is an area of 25-16-8+3=4 which has berries but not rocks.

11. Elevator

You are a fantastic trainer unbounded by common conventions, in fact you are so awesome you can have any number of Pokemon. This leads to some issues though. For example, since you wouldn't be able to carry so many Pokeballs they all must follow you in the real world at all times, just like Pikachu and Ash.
In your most recent journey you were faced with a problem. To get to the next gym you must climb a very steep cliff. Luckily, there is an elevator that you can use, it has some restrictions however. It is a special elevator with two shafts, one on the top and one on the bottom of the precipice. If the difference in weight in the shafts is less than a specific number defined for safety than you can activate the elevator and the shafts will change place.
Now the question is: given the safety restrictions on the elevator is it possible for you to take all your pokemons up the cliff using the elevator?
Notes: Your Pokemon are very organized and will follow your instructions perfectly. You can activate the elevator from the bottom and the top of the cliff or even from the inside.

Input Format

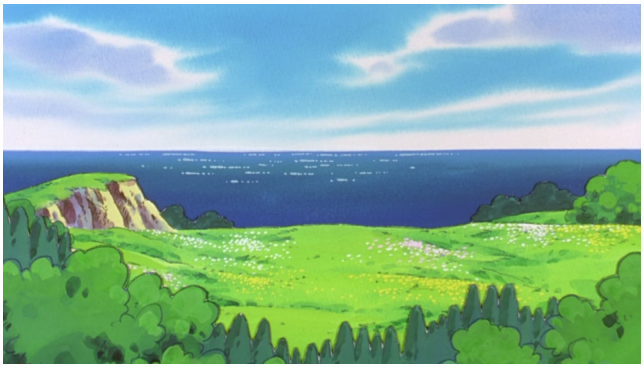
An integer `n` which is the number of test cases.
For each test case:
Two positive integers, the first one representing the number of pokemon you own and the second representing the maximum difference in weight between the shafts such that the elevator will operate safely.
A list of numbers representing the weight of each one of your Pokemon and you.
You may assume each test case contains at most $2 \cdot 10^5$ Pokemon.Instructions for use

Output Format

Output either "True" or "False"

Sample Input	Sample Output	Explanation
3 3 5 1 2 3 4 5 1 2 3 100 5 5 6 23 8 20 1	True False True	In the first test case you can transport each pokemon individually. In the second test case, to transport the pokemon with weight 100 you would need a counterweight of at least 95 which is much more than the total weight of all other pokemon. The third test case is more interesting and you should think about it by yourself.

12. Pokepark



You just bought a brand new plot of land around a beautiful lake. You are planning on building a park for your lovely Pokemon and to do so you must separate the plot in N different sections. Unfortunately, to separate the different sections you have to pay fees to the government. Whenever you separate a section of area A+B into one section of area A and another of area B you must pay a fee of A+B to the government. You can only separate a section in two each time. All sections must border the lake and the edge of the the plot, they must also be connected. This way, each section will always have two neighbors after the first two divisions.
Each final section borders the sections that appears before and after it in the list and the first and last sections border each other. All sections also border the lake and the border of the plot itself.
Given the areas of the sections you want to create, find out the minimum fee you will have to pay the government.

Input Format
An integer `n` which is the number of test cases.
For each test case: An integer representing the number of sections you want to divide the plot in and a list of integers representing the area of each of the sections of the plot we are going to build in order.

Output Format
An integer identifying the minimum amount of taxes that you can pay to build your park.

Sample Input	Sample Output	Explanation

`2`
`8`
`1 2 3 8 2 2 1 3`
`3`
`1 10 2`
	`61`
	`16`
		For the first test case we can make the separations as follows:
		`[[[[1 2] 3] 8][[[2 [2 1]] 3]]`
		For the second test case, the input is equivalent to `2 1 10`, so we can get `16` with `[[2 1] 10]`