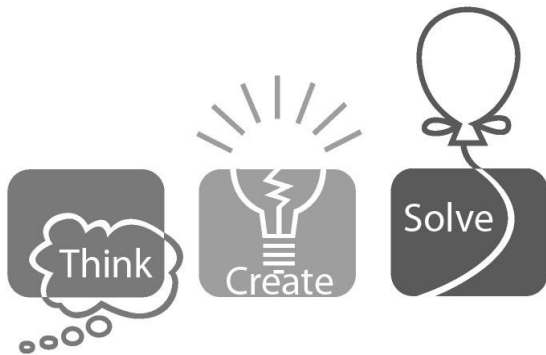


University of Central Florida



2024 (Fall) “Practice” Local Programming Contest

Problems			
Problem#	Difficulty Level	Filename	Problem Name
1	Easy	window	Window on the Wall
2	Easy	parity	Parity of Strings
3	Easy-Medium	remote	Historical TV Remote
4	Easy-Medium	sga	SGA President
5	Medium	soccer	World Cup Fever
6	Medium	rounding	Rounding Many Ways
7	Medium	circlesquare	Circle Meets Square
8	Medium-Hard	firstlast	First Last Sorting
9	Medium-Hard	shirts	Team Shirts/Jerseys

Call your program file: filename.c, filename.cpp, filename.java, or filename.py

For example, if you are solving World Cup Fever:

Call your program file: soccer.c, soccer.cpp, soccer.java, or soccer.py

UCF “Practice” Local Contest — August 19-23, 2024

Window on the Wall

filename: window

Difficulty Level: Easy

Time Limit: 5 seconds

Anya (Arup’s daughter) would like to add a window on the wall in her room. She asks Travis (the mathematician) to figure out the largest window she can have on her wall. Travis consults Chris (the engineer) to see if there are any structural constraints. Chris explains that there must be a minimum distance between the wall perimeter and the window perimeter for the wall to hold the window; otherwise the entire structure collapses.



The Problem:

Given the width and height of a rectangular wall and the window-border gap (minimum distance required between the perimeter of the wall and the perimeter of the window), determine the area of the largest rectangular window that can be installed on the wall.

The Input:

The input contains one line with three space-separated positive integers, w , h , and d ($w, h < 1000$, $d < 100$), representing, respectively, the wall’s width, wall’s height, and the minimum window-border gap amount needed.

The Output:

The output should be an integer on one line by itself, which represents the area of the largest rectangular window that can be installed. If it is not possible to install a window, output 0 (zero).

Sample Input

Sample Output

40 25 5	450
30 20 12	0
30 20 50	0
999 888 7	860890

UCF “Practice” Local Contest — August 19-23, 2024

Parity of Strings

filename: parity

Difficulty Level: Easy

Time Limit: 5 seconds

The historical battle between *numbers* and *strings* has taken a new twist: *numbers* are bragging on their categorization of being “even” or “odd” and *strings* lacking such feature. But don’t count *strings* out yet!

A string is considered “even” if *every* letter in the string appears an even number of times; the string is “odd” if *every* letter in the string appears an odd number of times.

The Problem:

Given a string, determine whether the string is even, odd, or neither.

The Input:

The input consists of a single line, starting in column 1, not exceeding column 70, and containing only the lowercase letters (at least one letter).

The Output:

The output consists of a single integer: print 0 (zero) if the string is even, 1 (one) if the string is odd, or 2 if the string is not even and is not odd (i.e., it is neither).

Sample Input

Sample Output

coachessoahwwwww	0
coachesarefun	2
coachesc	1

UCF “Practice” Local Contest — August 19-23, 2024

Historical TV Remote Control

filename: remote

Difficulty Level: Easy-Medium

Time Limit: 5 seconds

As Dr. Orooji is getting older, he is becoming more attached to older items and has difficulty letting go of them (he claims they have historical value). For example, he still has the first table he got for the programming team! The situation is the same at home, e.g., there is a broken TV remote control but Dr. O still uses it, because he considers it an old item with historical value!

The Problem:

The old remote control has 12 buttons: digits 0-9, channel down, and channel up. There are no other buttons on the remote control. Some digits on the remote don’t work but channel up/down always works. So, to get to a particular channel, Dr. O sometimes has to use the channel up/down. For example, let’s assume digits 0 and 5 on the remote don’t work:

If Dr. O wants to watch channel 102, he would select 99 and then “channel up” 3 times.

If he wants to watch channel 597, he would select 611 and then “channel down” 14 times.

Given the digits that do not work and a target channel, determine how many times Dr. O needs to hit channel up or down. Dr. O, of course, wants to exert the least energy, hence he wants to hit the channel up/down the minimum number of times. Assume that Dr. O will enter a channel between 0 and 999 (inclusive) to start and that channel down has no effect at 0 and channel up has no effect at 999.

The Input:

The first input line contains an integer, n ($1 \leq n \leq 9$), indicating how many digits on the remote do not work. These broken digits are listed (in increasing order) on the same input line. The second input line provides the target channel (an integer between 1 and 999, inclusive).

The Output:

The output consists of a single integer, indicating how many times Dr. O needs to hit channel up/down. Note that, since one or more digits work, it is always possible to reach the target channel.

Sample Input

Sample Output

3 0 8 9 35	0
4 1 2 5 9 250	50

UCF “Practice” Local Contest — August 19-23, 2024

SGA President

filename: sga

Difficulty Level: Easy-Medium

Time Limit: 2 seconds

After an amazing performance at World Finals, Timothy and Alex, who are no longer eligible for ICPC, have decided to look for a new challenge. They'd like to run for SGA President and Vice President. Unfortunately, they have realized that with tickets from the previous election such as Josh/Jad and Brad/Breon, they have no hope of winning because all winning tickets must have two distinct names that start with the same first letter, so Timothy and Alex just won't do.

Naturally, Timothy was despondent about this revelation and to make himself feel better came up with a problem for locals. Given the names of each student at UCF, Timothy wondered how many potential winning pairs for SGA President and Vice-President there might be. In order for a pair to have the potential to win, their names must be different but start with the same first letter. Since President and Vice President are different roles, we count the ticket of Josh and Jad differently than the ticket of Jad and Josh. (The first name listed is the candidate for President while the second name listed is the corresponding candidate for Vice President.) Note that UCF has many students that share a first name, so there might be several potential winning pairs of Josh and Jad. For example, if there are 10 Joshes and 3 Jads on campus, there are 30 Josh/Jad pairs with a Josh running for President and a Jad running for Vice President (and these should all be counted).

The Problem:

Given the names of each UCF student, calculate the number of possible President/Vice-President pairs who have a potential to win the SGA election.

The Input:

The first line of input contains a single positive integer, n ($n \leq 66,183^1$), representing the number of UCF students. The following n lines each contain a single first name of one UCF student. All names will consist of uppercase letters only and be between 1 and 20 letters long, inclusive. Each line represents a distinct student, but distinct students may have the same first name.

The Output:

On a line by itself, output the total number of President-Vice President pairs that have a chance to win the SGA election.

¹ This was the actual enrollment at UCF in the 2017 fall semester!

Sample Input**Sample Output**

10 JOSH JAD JENNIFER JENNIFER JALEN HASAAN ALI TIM ALEX TRAVIS	22
5 ALEX BRANDY CELINE DWAYNE ELIZABETH	0

UCF “Practice” Local Contest — August 19-23, 2024

World Cup Fever

filename: soccer

Difficulty Level: Medium

Time Limit: 2 seconds

The 2018 World Cup was held in Russia. Some great soccer countries (e.g., Italy, Netherlands, Chile, USA) did not qualify for this World Cup. These countries found out that they needed more effective passing.

The Problem:

Given the player positions for two teams, determine the minimum number of passes needed to get the ball from one player to another player. For the purposes of this problem, players do not change position, i.e., they do not move.

Player P_1 can pass the ball directly to P_2 if they are on the same team and no other player is in between the two players.

Let's assume:

- P_1 and P_2 are on the same team
- P_1, P_2, P_3 form a line with P_3 between P_1 and P_2
- There are no other players on the line formed by P_1, P_2, P_3

Then,

- If P_3 is on the other team, P_1 cannot pass the ball directly to P_2 .
- If P_3 is on the same team, P_1 can pass the ball to P_3 to pass it to P_2 .

The Input:

The first input line contains an integer, n ($2 \leq n \leq 11$), indicating the number of players on each team. The second input line contains $2n$ integers, providing the (x,y) coordinates for the n players on Team 1; the first integer on this input line is the x coordinate for Player 1, the second integer is the y coordinate for Player 1, the third integer is the x coordinate for Player 2, etc. The third input line provides (in a similar fashion) the (x,y) coordinates for the n players on Team 2. Assume that all coordinates are integers between 1 and 999 (inclusive) and that all players are on distinct locations, i.e., no two players occupy the same spot (point).

Assume Player 1 on Team 1 has the ball and wants to pass the ball to Player n on Team 1. Assume that any player can pass the ball any distance.

The Output:

The output consists of a single integer, indicating the minimum number of passes needed to get the ball from Player 1 on Team 1 to Player n on Team 1. If it is not possible to get the ball from Player 1 to Player n , print -1 .

Sample Input**Sample Output**

3 10 15 13 17 10 19 10 17 16 17 13 19	2
5 1 1 3 1 5 1 7 1 9 1 2 1 4 1 6 1 8 1 10 1	-1
3 1 1 5 5 2 2 10 10 50 50 20 20	1

UCF “Practice” Local Contest — August 19-23, 2024

Rounding Many Ways

filename: rounding

Difficulty Level: Medium

Time Limit: 2 seconds

Timothy and Alex’s hopes and dreams of running for UCF’s Student Government Association have been crushed by the realization that their campaign ticket would not be alliterative. So, they have decided to analyze statistics from many different polls given to students to determine what pair of programming team members would be best situated to win the election. However, there is a problem. All of the statistics have been rounded off. This would not be an issue apart from the fact that the pollsters forgot to mention how the number was rounded!

(Math Terminology Note: if we round, say, 198 to 200, then 198 is called the “true value” and 200 is called the “rounded value”.)

For example, the rounded value 750 could have come from a true value rounded to the nearest 10 or maybe even the nearest 250. It may also have come from a true value rounded to the nearest 1 (thus not rounding it at all). Thus, the true value could have been something like 625 or maybe much closer like 748.

Luckily for Timothy and Alex, after some reconnaissance work, they have discovered the general rounding methods used:

- The original statistic was a positive integer S
- Then a positive integer X was chosen such that X is a divisor of some power of 10, i.e., there exist non-negative integers Y and Z such that $X * Y = 10^Z$
- Finally the statistic S was rounded to the nearest positive multiple of X to get the rounded value N , i.e., there exists a positive integer W such that $X * W = N$ and $|S - N|$ is minimized.

The Problem:

Given the rounded value, find all the different ways it could have been rounded (derived). In other words, given N and using the above constraints, you are to find all values of X that satisfy both of the following two equations:

- $X * Y = 10^Z$
- $X * W = N$

The Input:

The first and only line of input contains an integer, N ($1 \leq N \leq 10^{18}$), representing the rounded value.

The Output:

First print out a single line containing an integer representing the number of different **X** values that the rounded value **N** could have been derived from. Then print out all of these values of **X**, in increasing order, on separate lines.

Sample Input

Sample Output

30	4 1 2 5 10
120	8 1 2 4 5 8 10 20 40
8	4 1 2 4 8

Explanation for the first Sample Input/Output:

Output: 1

$X = 1, Y = 10, Z = 1, W = 30$

Rounded to nearest multiple of 1: $1*10=10, 1*30=30$

Output: 2

$X = 2, Y = 5, Z = 1, W = 15$

Rounded to nearest multiple of 2: $2*5=10, 2*15=30$

Output: 5

$X = 5, Y = 2, Z = 1, W = 6$

Rounded to nearest multiple of 5: $5*2=10, 5*6=30$

Output: 10

$X = 10, Y = 1, Z = 1, W = 3$

Rounded to nearest multiple of 10: $10*1=10, 10*3=30$

UCF “Practice” Local Contest — August 19-23, 2024

Circle Meets Square

filename: circlesquare

Difficulty Level: Medium

Time Limit: 2 seconds

We all know that you can’t put a round peg in a square hole. Asking you to do so in this contest would be cruel and unusual punishment, which is banned by the Eighth Amendment to the United States Constitution. But, perhaps a more reasonable problem that the Framers of the Constitution never thought about is determining if a given circle and square have an intersection of positive area (overlap), or touch (share a point in common), or don’t touch at all.

The Framers of the US Constitution and the UCF Programming Team coaches would like to know, given a circle and a square, do the two overlap in some positive area, touch, or don’t touch at all. Help them out by writing a program to solve the problem!

The Problem:

Given the description of a square and circle in the Cartesian plane, determine if the intersection between the two has positive area (overlap), is a single point (touches) or doesn’t exist.

The Input:

The first input line contains three integers: x ($-1,000 \leq x \leq 1,000$), y ($-1,000 \leq y \leq 1,000$), and r ($0 < r \leq 1,000$), representing (respectively) the x and y coordinates and radius of the circle.

The second input line contains three integers: t_x ($-1,000 \leq t_x \leq 1,000$), t_y ($-1,000 \leq t_y \leq 1,000$), and s ($0 < s \leq 1,000$), where (t_x, t_y) represents the coordinates of the bottom left corner of the square and s represents the side length of the square. The square’s top right corner is (t_x+s, t_y+s) , so that its sides are parallel to the x and y axes.

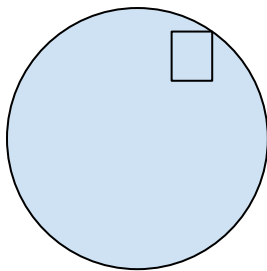
The Output:

If the circle and square *don’t touch*, output 0 (zero). If they *touch* at a single point, output 1 (one). If they *overlap* in positive area, output 2 (two).

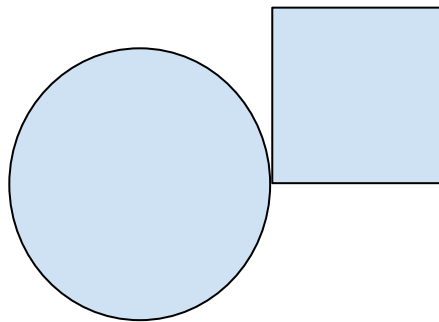
Sample Input**Sample Output**

0 0 5 2 3 1	2
0 0 5 5 0 6	1
0 5 4 -1 -1 1	0

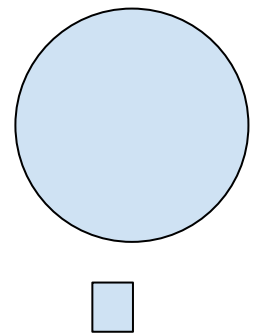
Pictures of the Sample Input:



First Sample



Second Sample



Third Sample

UCF “Practice” Local Contest — August 19-23, 2024

First Last Sorting

filename: firstlast

Difficulty Level: Medium-Hard

Time Limit: 2 seconds

Arup has just created a data structure that makes the two following list transformations in constant $O(1)$ time:

- Take any element in the list and move it to the front.
- Take any element in the list and move it to the back.

You've realized that sorting speed can be improved using these transformations. For example, consider the input list:

8, 3, 6, 7, 4, 1, 5, 2

We can do the following sequence of transformations to sort this list:

8,	3,	7,	4,	1,	5,	2,	6	(move 6 to end)
8,	3,	4,	1,	5,	2,	6,	7	(move 7 to end)
2,	8,	3,	4,	1,	5,	6,	7	(move 2 to front)
1,	2,	8,	3,	4,	5,	6,	7	(move 1 to front)
1,	2,	3,	4,	5,	6,	7,	8	(move 8 to end)

You are now curious. Given an input array of distinct values, what is the fewest number of these first/last operations necessary to sort the array?

The Problem:

Given an initial permutation of the integers 1, 2, ..., n , determine the fewest number of first/last operations necessary to get the list of values sorted in increasing order.

The Input:

The first line of input will contain a single positive integer, n ($n \leq 10^5$), representing the number of values to be sorted. The next n lines contain one integer each. All of these integers will be distinct values in between 1 and n (inclusive), representing the original order of the data to sort for the input case.

The Output:

On a line by itself, output the fewest number of first/last operations necessary to sort the input list.

Sample Input**Sample Output**

8 8 3 6 7 4 1 5 2	5
5 1 2 5 3 4	1

UCF “Practice” Local Contest — August 19-23, 2024

Team Shirts/Jerseys

filename: shirts

Difficulty Level: Medium-Hard

Time Limit: 2 seconds

The Legendary Mathematician Travis and his friends are in a recreational (rec) league for competitive programming. Travis has n friends and his friends want to show a sense of unity for this competitive programming league, so they splurged on team shirts/jerseys each of which has an integer between 1 and 99 (inclusive) on the back. Each of Travis’s friends have already selected their own (not necessarily distinct) jersey number so we have a list of n integers. Travis now needs to choose his jersey number to complete a list of $n+1$ integers. Travis will also choose an integer between 1 and 99 (inclusive) and he does not have to pick a number different from what his friends have picked (i.e., he can choose to duplicate a jersey number).

Even though Travis can choose any jersey number, he wants to show why he is considered to be a legendary mathematician. Travis has a favorite positive integer and he wants to select his jersey number such that he would be able to pick a set of numbers from the list of $n+1$ integers, concatenate this set of numbers together, and reach his favorite integer without extra leading or trailing digits (since Travis wants his favorite integer and not some other integer, he is trying to do so without any extra trailing or leading digits).

Travis has the fortunate option to choose his jersey number last. Now he just needs to determine if he can select some number that guarantees he can form his favorite integer. For example, suppose Travis’s friends have selected the jersey numbers 3, 10, 9, and 86. Then, by selecting the number 75, Travis could form his favorite positive integer 8,675,310. Note that Travis didn’t need to use the jersey number 9.



Note that if Travis chooses to use a jersey number, he must use the number completely, i.e., he cannot use just some of the digits in the number. For example, if he decides to use 75 in the above example, he must use “75”; he cannot just use “7” or use just “5”. Note also that if he wants to use a particular jersey number multiple times, he must have multiple friends with that jersey number. For example, if he wants to use 75 multiple times, he must have multiple friends with the number 75 (he can, of course, pick 75 for himself as well to increase the number of occurrences of 75 in the list of $n+1$ numbers).

We now have to break the secret that Travis is very temperamental! He tends to gain and lose friends very easily (so much for that whole unity thing); he also changes his favorite integer more often than the lights change at a busy intersection. For these reasons, Travis needs your help to write a program to solve this problem for a general set of friends and favorite numbers.

The Problem:

Given a set of positive integers representing friends' jersey numbers, and a favorite integer, determine if Travis can choose a jersey number (for himself) that can allow for the creation of his favorite integer via concatenation of zero or more friends' numbers and potentially his number. Additionally, since this is a rec league, the list of friends may contain duplicate jersey numbers, and Travis can choose an already chosen number for his shirt.

The Input:

The first line of input contains exactly one positive integer, t ($t < 1,000,000,000$), representing Travis's favorite integer. The second line of input contains exactly one positive integer, n ($n \leq 25$), representing how many friends Travis has today. The next input line contains n space separated integers which denote the jersey number chosen by each of Travis's friends. The jersey numbers will be between 1 and 99 (inclusive) and will have no leading zeroes, e.g., the input would have jersey number 7 but not 07.

The Output:

If Travis is capable of reaching his favorite integer with (or without) his set of friends, print 1 (one); otherwise output 0 (zero).

Sample Input

Sample Output

707 2 7 24	1
70707 2 7 7	0
1122 3 21 1 23	0
715 1 75	0