

University of Central Florida



2026 Local
Programming
Contest
(Final Round)

Problems			
Problem#	Difficulty Level	Filename	Problem Name
1	Easy	bulk	More and Cheaper!
2	Easy	dream	Contest Score Dream
3	Easy	pairs	Letter Pairs
4	Easy-Medium	digits	Multiply My Digits
5	Easy-Medium	chargeon	Charge On!
6	Medium	necklaces	Counting Necklaces
7	Medium	seamonsters	Scylla and Charybdis and Many More!
8	Medium	battle	Battle of Wills
9	Medium	coins	Intergalactic Currencies
10	Medium-Hard	bricks	Tomas's Bricks
11	Medium-Hard	ymtd	You Make the Data!
12	Medium-Hard	diamond	Diamond Cutting
13	Hard	sleeper	Sleeper Agents
14	Hard	np	Not another Polyomino!

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

For example, if you are solving Intergalactic Currencies:

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

UCF Local Contest (Final Round) — September 12, 2026

More and Cheaper!

filename: bulk

Difficulty Level: Easy

Time Limit: 5 seconds

Most companies selling merchandise in bulk have one price per item if you order less than X items and a lower price per item if you order X (or more) items. Sometimes, these companies are not careful and it is actually cheaper to order X items even if you need fewer. For example, they say \$3 per item if ordering less than 100 and \$2 per item if ordering 100 or more. So, in this scenario, if you need only 80 of the item, it is cheaper to order 100 (and just give away the extra ones) as opposed to ordering 80 that you need (\$200 vs. \$240).

The Problem:

Given the price breakdown for an item and the quantity you need, determine if you should order exactly what you need or if it is better to order the larger quantity (and just give away the extra ones).

The Input:

There is only one input line; it provides four integers in the following order:

- value for X ,
- price per item if ordering $\geq X$,
- the quantity you need, and
- price per item if ordering $< X$.

Assume that each input integer is between 1 and 10^3 (inclusive), that price per item if ordering $\geq X$ is lower than price per item if ordering $< X$, and that the quantity you need is $< X$.

The Output:

Print the word `need` if you should order exactly what you need. Print the word `bulk` if it is better to order the larger quantity. If the total cost is the same both ways, order X (i.e., get the extra items).

Sample Input

Sample Output

100 2 80 3	bulk
200 4 30 7	need
1000 10 500 20	bulk

UCF Local Contest (Final Round) — September 12, 2026

Contest Score Dream

filename: dream

Difficulty Level: Easy

Time Limit: 5 seconds

You were too excited the night before the local contest and could not fall asleep. You finally fell asleep and had this good dream of seeing how long (in minutes) it takes you to solve each contest problem. Your dream was about to show you if you won the contest but, unfortunately, you woke up! Well, the coaches decided to include this problem in the contest to make sure you remember the order to solve the problems in the contest that will result in the most problems solved in contest time with the least total penalty points.

The Problem:

Given the number of problems in the contest and how long (in minutes) it takes you to solve each problem, determine the most problems that you can solve with the least total penalty points. Assume a 5-hour (300-minute) contest.

The Input:

The first input line contains an integer, p ($1 \leq p \leq 15$), indicating the number of problems in the contest. Each of the following p input lines contains an integer (between 1 and 300, inclusive) indicating how long it will take you to solve the given problem.

The Output:

Print the number of problems solved and total penalty points if problems are solved in the optimal order. Note that:

- Only one problem is solved at a time, i.e., serial (not parallel) processing.
- Penalty points (for time) for each problem is computed from the contest start time and not since the time you finished your last problem.
- For each solved problem, you get one penalty point per minute.
- There may not be enough time in the contest to solve all the problems.

(Sample Input/Output on the next page)

Sample Input**Sample Output**

2 200 100	2 400
3 100 100 100	3 600
5 200 60 30 150 100	3 310

Explanation of Sample Input/Output:

In the first Sample, there is time to solve both problems. If you solve Problem #1 followed by Problem #2, the penalty points for Problem #1 is 200 and the penalty points for Problem #2 is 300, resulting in 500 total penalty points. If you solve Problem #2 followed by Problem #1, the penalty points for Problem #2 is 100 and the penalty points for Problem #1 is 300, resulting in 400 total penalty points.

In the third Sample, there is not enough contest time to solve more than 3 problems.

UCF Local Contest (Final Round) — September 12, 2026

Letter Pairs

filename: pairs

Difficulty Level: Easy

Time Limit: 5 seconds

In English text, letters 'E' and 'T' occur the most. But after the letter Q, letter 'U' occurs more than any other letter.

The Problem:

Given a string containing only lowercase letters and blanks, for each letter in the string, print the letter that occurs the most immediately after it. Note that blanks separate words so two letters are not adjacent if there are blanks (one or more blanks) between the two letters.

The Input:

There is only one input line; it provides the string to be analyzed. Assume that:

- the string contains only lowercase letters and blanks (i.e., no other characters),
- the total length for the string does not exceed 70 characters,
- the string starts with a letter and ends with a letter (i.e., no blanks at the beginning/end of the string), and
- the string contains at least one letter.

Note that there may be several blanks (i.e., not exactly one blank) between the words in the string.

The Output:

Print the letters in the string in alphabetical order, one letter per output line.

For each letter in the string, print the letter that occurs the most immediately after it.

If multiple letters occur the most for a given letter, print them in alphabetical order.

If a letter in the string does not have any letter after it (e.g., only blanks after it and/or end of string after it), print '?' for it.

Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input**Sample Output**

abd deyz abc bg pqpq pppp st	a: b b: c d g c: ? d: e e: y g: ? p: p q: p s: t t: ? y: z z: ?
abacaa	a: a b c b: a c: a
z	z: ?

Notes:

- In the first Sample Input/Output, the letter 'q' appears twice immediately after 'p' but the letter 'p' appears three times immediately after 'p', so the letter that occurs the most immediately after 'p' is 'p'. Also note that this sample illustrates that two letters are not adjacent if there are blanks between them, e.g., we don't consider 'a' is adjacent to 'z' because there is a space between them.
- An output line starts with a letter followed by a colon (':'), followed by exactly one space, followed by either a question mark ('?') or a set of letters separated from each other by exactly one space.

UCF Local Contest (Final Round) — September 12, 2026

Multiply My Digits

filename: digits

Difficulty Level: Easy-Medium

Time Limit: 5 seconds

The USA celebrated its 250th birthday this past July. The number 250 has several properties, one of which is that it can be produced by multiplying some (or all) of its digits, namely, $250 = 2 \times 5 \times 5 \times 5$. Below are a few more numbers with this property:

$$36 = 6 \times 6$$

$$64 = 4 \times 4 \times 4$$

$$175 = 7 \times 5 \times 5$$

$$735 = 7 \times 7 \times 3 \times 5$$

The Problem:

Given a number, determine whether or not it can be produced by multiplying some (or all) of its digits.

The Input:

The first input line contains an integer, n ($1 \leq n \leq 20$), indicating the number of values to be analyzed. Each of the following n input lines contains an integer, v ($10 \leq v \leq 10^6$), to be analyzed.

The Output:

For each integer to be analyzed, print 1 (one) if the number can be produced by multiplying some (or all) of its digits; 0 (zero) otherwise.

Sample Input

Sample Output

3	1
250	0
1000000	1
384	
4	1
25	0
27	0
43	1
875	

UCF Local Contest (Final Round) — September 12, 2026

Charge On!

filename: chargeon
Difficulty Level: Easy-Medium
Time Limit: 3 seconds

Knightro uses his phone quite a lot and has developed a consistent routine of phone usage per day. The routine consists of a sequence of activities, where each activity has a charge delta x . That is, the activity adds x to the phone's charge, capped by its battery capacity, b . A positive value of x corresponds to Knightro charging his phone; a negative value of x corresponds to Knightro using his phone. That is, if Knightro's current charge on his phone is c , after performing an activity with charge delta x , the new charge of Knightro's phone will be equal to the minimum of b and $c + x$. Note that x will never equal zero.

At the beginning of his routine, Knightro's phone is fully charged to the battery's maximum capacity. As previously mentioned, the battery can never go above its capacity, even while charging. If the battery level ever reaches zero or below, the phone dies, which Knightro would like to avoid.

The Problem:

Determine the minimum battery capacity that allows Knightro to make it through his entire routine without his phone dying.

The Input:

The first input line contains an integer, n ($1 \leq n \leq 10^5$), representing the number of activities in the routine. The next input line contains n non-zero integers representing the activities in order; each activity's charge delta will range from -10^4 to 10^4 , inclusive.

The Output:

Output a line containing a single integer representing the minimum battery capacity the phone would need to complete the routine without dying. It can be shown that the maximal output will not exceed the maximum value of a standard signed 32-bit integer ($2^{31} - 1$).

Sample Input

Sample Output

4 -5 -4 3 -7	14
5 1 2 3 4 -10	11

UCF Local Contest (Final Round) — September 12, 2026

Counting Necklaces

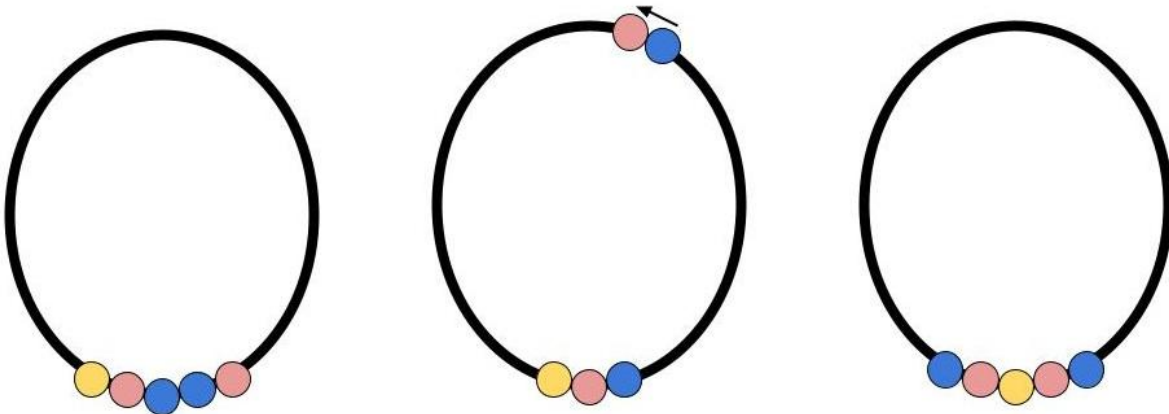
filename: necklaces

Difficulty Level: Medium

Time Limit: 3 seconds

Her Royal Majesty of the Luminous Pearl Commonwealth has tasked you, her famed royal jeweler, with designing a magnificent necklace for her. She has told you that the necklace must consist of three or more of the commonwealth's precious-colored pearls, and the number of the pearls on the necklace must be odd (i.e., 3, 5, 7, ...). The queen, however, believes true beauty lies in symmetry, so the sequence of pearl colors must be the same when read forwards or backwards. That is, the sequence of pearls must be an *odd-length palindrome*.

As the pearls are quite rare and valuable, she has given you a string of colored pearls. To satisfy her request, you can first cut out a contiguous subsequence of the pearls (zero or more pearls) from the left end and cut out a contiguous subsequence of the pearls (zero or more pearls) from the right end. Then you will rotate the remaining sequence of pearls by some non-zero amount. The three images below illustrate how a sequence may be rotated to achieve a valid necklace (note that the image on the left is not a palindrome but the image on the right is):



To describe the process more formally, you can choose three values: L , R , and k , such that:

- $1 \leq L \leq R \leq n$,
- $R - L + 1$ is odd and at least 3, and
- $1 \leq k \leq R - L$.

You will then remove all pearls before position L and all pearls after position R . Then, on the remaining pearls, you will take the last pearl and move it to the front of the sequence k times, i.e., you will cyclically rotate the sequence to the right k times. You now want to determine how many ways you can pick values of L , R , and k such that the resulting sequence of pearls results in an odd-length palindrome. Note that k must be non-zero, i.e., the remaining pearls must be cyclically shifted at least once.

The Problem:

Given a sequence of n colored pearls, count the number of ways of choosing three values (L , R , and k) such that cyclically rotating the contiguous subsequence of pearls from L to R (inclusive) to the right by k results in the colors of that subsequence being an *odd-length palindrome*.

The Input:

There is only one input line; it consists of a string of lowercase English letters, corresponding to the initial sequence of colors of the given pearls. The given string will contain at least 3 letters and at most 5000 letters.

The Output:

Output a single integer corresponding to the number of ways to choose the values L , R , and k .

Sample Input Sample Output

ypbbp	3
abaccaba	8
abc	0

Note: For the first Sample Input, the three ways to choose values are:

$L = 1, R = 5, k = 2$, resulting in odd-length palindrome bpy**p**b
 $L = 2, R = 4, k = 1$, resulting in odd-length palindrome b**p**b
 $L = 3, R = 5, k = 2$, resulting in odd-length palindrome b**p**b

UCF Local Contest (Final Round) — September 12, 2026

Scylla and Charybdis and Many More!

filename: seamonsters

Difficulty Level: Medium

Time Limit: 3 seconds

Odysseus is searching for a way home and has to travel through a sea filled with monsters. The sea can be modeled by $n-1$ passages (that can be traversed in either direction) connecting n regions, such that all pairs of regions can directly or indirectly reach each other. Regions are numbered 1 through n , and Odysseus starts in region 1. For every region r , there is no sequence of passages, where Odysseus starts in r , uses each passage exactly once, and returns to region r . Odysseus is currently in the first region, and there are several exits for escaping the sea.

The odyssey, however, is not that simple. Odysseus has angered Poseidon, God of the seas; thus, Poseidon will deploy several sea monsters, such as Scylla and Charybdis, to try to destroy Odysseus's ship. In one unit of time, the following sequence of events occurs in order:

1. Poseidon chooses a passage in which to deploy a sea monster. The sea monster will stay in the passage indefinitely.
2. If Odysseus is currently in region r , he chooses a region directly connected to r and travels to it using the passage connecting the two regions. Odysseus knows the location of all deployed sea monsters.

If at any point Odysseus uses a passage with a sea monster, his ship will be destroyed and cannot escape. At any point in time, if Odysseus is in a region with an exit, he escapes the sea. Assuming Odysseus knows the layout of the sea's passages and regions, can Odysseus guarantee his escape, regardless of Poseidon's choices?

The Problem:

Given the network of passages and regions, can Odysseus reach some exit without his ship being destroyed if he starts at region 1?

The Input:

The first input line consists of two integers, n and m ($2 \leq n \leq 10^5$, $1 \leq m \leq n$), corresponding to the number of regions and the number of exits, respectively.

The following input line will consist of m integers between 1 and n corresponding to which regions contain an exit. It is guaranteed that all given exits are unique. Note that any region can be an exit and if Odysseus is in a region with an exit, he escapes the sea.

Each of the following $n-1$ input lines consists of two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), denoting a passageway between regions u and v . It is guaranteed that all pairs of regions are directly or

indirectly connected. Note that the input passages are not listed in any particular order and that the two regions for a passage are not listed in any particular order.

The Output:

Print YES or NO, indicating whether or not Odysseus can escape the sea.

Sample Input Sample Output

3 1 3 1 2 2 3	NO
6 3 4 5 2 1 2 1 3 3 4 3 5 1 6	YES
6 2 4 5 1 2 1 3 3 4 3 5 4 6	NO

UCF Local Contest (Final Round) — September 12, 2026

Battle of Wills

filename: battle

Difficulty Level: Medium

Time Limit: 3 seconds

Will doesn't like to lose. One night, when playing pool with Joe, he lost his first match, to which Will asked Joe, “how about best of 3?” Unfortunately for Will, Joe was pretty good at pool and became the first player with 2 victories, the first player with 3 victories, etc. as Will kept on asking to extend the series. Finally, after Will asked Joe, “how about best of 39?” and Joe said yes (it takes two to make this a serious problem), Will was able to close the deal and win the series 20 - 19.

One might wonder, given a best of $2n - 1$ series, where the winner is the first person to win exactly n games, how many different ways can the games go so that the eventual winner would have lost all of the smaller best of series (best of 1, best of 3, best of 5, ..., best of $2n - 3$), but won the overall best of $2n - 1$ series?

For example, let our teams be called A and B. With $n = 2$, we are looking for all possible ways in which a team could win a best of 3 series while losing the first game, so the sequences of games to count are ABB and BAA. For $n = 3$, we are looking for all of the possible ways in which a team could win a best of 5 series but lose the prefix best of 1 and best of 3 series. These possibilities are AABBB, ABABB, BABAA and BBAAA.

The Problem:

Given a positive integer n , determine the number of sequences of best of $2n - 1$ game series which have the winning team losing all of best of smaller series (1, 3, 5, ..., $2n - 3$).

The Input:

There is only one input line; it contains a single integer, n ($1 \leq n \leq 1000$), indicating the number of games the winning team must win to win the series.

The Output:

Print, on a line by itself, the number of sequences of games that satisfy the criteria. Since the answer could be very large, print this value modulo $10^9 + 7$.

(Sample Input/Output on the next page)

Sample Input**Sample Output**

1	2
2	2
3	4

UCF Local Contest (Final Round) — September 12, 2026

Intergalactic Currencies

filename: coins

Difficulty Level: Medium

Time Limit: 3 seconds

Apollo the space pup is travelling across the universe through many galaxies. As he visits various planets, he encounters many alien civilizations where he must use different currencies. An alien currency consists of n types of coins, each with a distinct value, and there will always be a coin type with value 1.

As he cannot communicate with these alien species (he has a hard enough time communicating with Earthlings!), he wants to be prepared to make a single purchase; however, he does not know the exact value of the purchase, only that it will be an integer value between 1 and k , inclusive. To ensure that he can make the purchase without carrying an excess amount of coins, he wants to know the minimum number of coins he needs such that he can make any single purchase of an integer amount between 1 and k (inclusive). Apollo does not mind having multiple of the same type of coin, as long as the total number of coins is minimal.

The Problem:

Given the types of coins in the alien currency and the maximum potential purchase Apollo wants to make, determine the minimum number of coins Apollo needs such that he can make a single purchase of any value between 1 and k , inclusive. Note that, although Apollo does not plan to make any purchases with price greater than k , the sum of his coins may exceed k .

The Input:

The first input line contains two integers, n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq 10^{18}$), representing (respectively) the number of coin types available and the maximum potential purchase that needs to be covered. The next input line contains n distinct integers (each between 1 and 10^{18} , inclusive) representing the values of each coin type. There will always be a unit of 1.

The Output:

Output a line containing a single integer representing the minimal number of coins needed to cover any potential purchase of price 1 to k , inclusive.

(Sample Input/Output on the next page)

Sample Input**Sample Output**

3 6 1 5 2	4
2 10 1 5	6

Note:

To cover any purchase with a value between 1 and 6 (inclusive) in the first sample, one valid list of coins is $\{1, 2, 2, 5\}$. It can be shown that 3 or fewer coins cannot cover all possible required purchases with the given coin types.

UCF Local Contest (Final Round) — September 12, 2026

Tomas's Bricks

filename: bricks

Difficulty Level: Medium-Hard

Time Limit: 3 seconds

Tomas is going on a trip to Europe! His trip to Europe consists of flights with different airlines, and each airline has its own weight limit for suitcases. Tomas has a beloved brick collection, and he wants to bring as many bricks as he can on his trip to Europe (don't ask *why* Tomas is trying to smuggle bricks into Europe). Every brick has an integer weight assigned to it, and Tomas's brick collection is constantly changing, with bricks entering and exiting the collection all the time. While Tomas deals with his changing brick collection, help him determine, for each weight limit, how many bricks he can take on the trip.

The Problem:

Given a series of updates to Tomas's brick collection, process queries to determine the maximum number of bricks he can pack for a specific flight weight limit.

The Input:

The first input line contains two integers: n ($1 \leq n \leq 10^5$), indicating the number of bricks initially in Tomas's collection, and q ($1 \leq q \leq 10^5$), indicating the number of queries. The next input line contains n integers, the weights of the n bricks currently in Tomas's collection. Each of the next q input lines contains a query to be processed. There will be three types of queries:

- Add a brick: This input line starts with the letter A, followed by a space, followed by a number representing a weight. Tomas will add a brick with that weight to his collection.
- Remove a brick: This input line starts with the letter R, followed by a space, followed by a number representing a weight. Tomas will remove a brick with that weight from his collection. It is guaranteed that a brick with that weight is in Tomas's collection before the query.
- Weight limit: This input line starts with the letter W, followed by a space, followed by a number representing a weight limit. This query is asking for the maximum number of bricks from Tomas's collection that he can put in his suitcase with the given weight limit.

Every brick will have a weight between 1 and 10^5 (inclusive), and every weight limit will be between 1 and 10^{10} (inclusive). Tomas's brick collection may contain multiple bricks with the same weight (note that the R query removes only one brick even if there are multiple bricks with the given weight).

The Output:

There is no output required for the queries that add or remove bricks. For each weight limit query, output a separate line providing the maximum number of bricks Tomas can put in his suitcase from his collection at that current point. Note that the collection will not have more than 2×10^5 bricks.

Sample Input

Sample Output

3 3 2 2 4 W 4 R 2 W 4	2 1
6 7 10 5 8 4 2 3 W 9 W 8 W 29 R 2 W 29 A 8 W 29	3 2 5 4 5

UCF Local Contest (Final Round) — September 12, 2026

You Make the Data!

filename: ymtd

Difficulty Level: Medium-Hard

Time Limit: 1 second

A positive integer v is *sturdy* if $2 \times v$ is a derangement of the same digits as the original value v . In other words, $2 \times v$ consists of the same digits as v , rearranged in a different order such that, for each position, the digits in v and $2 \times v$ differ. (Note that the frequencies of each digit must match, e.g., if there are, say, five occurrences of the digit 7 in v , then there must be five occurrences of the digit 7 in $2 \times v$ as well.)

Normally, this is where we might ask you to determine if an integer v is *sturdy* or not, but unfortunately we lost all of the test cases for the original problem and need your help!

Your mission, should you choose to accept it, is to find *sturdy* numbers of varying lengths.

The Problem:

Given an integer n , find any n -digit sturdy number v such that v and $2 \times v$ are derangements of each other, or report that none exist.

The Input:

There is only one input line; it contains a single integer, n ($1 \leq n \leq 2 \times 10^5$), representing the required number of digits for the sturdy number.

The Output:

If there exists no n -digit sturdy number, print -1. Otherwise, print a single line containing any positive n -digit sturdy number. The output should not include leading zeros.

Sample Input

Sample Output

2	-1
19	3141592658378236746

Note: 3141592658378236746 is sturdy because when it is multiplied by 2, you get 6283185316756473492, which consists of the same digits in a different order and there are no position with the same digit in both numbers.

UCF Local Contest (Final Round) — September 12, 2026

Diamond Cutting

filename: diamond

Difficulty Level: Medium-Hard

Time Limit: 4 seconds

You recently got scammed on the popular e-commerce site Sahara, which has recently been overwhelmed by an influx of AI sellers and buyers. When you bought “sparkly compressed carbon gems,” you were convinced you were buying several diamonds for two dollars! Instead, you got several lumps of coal in the shape of convex quadrilaterals.

After leaving a scathing review, you try to resell the coal using the same technique, but all of the AI buyers have learned from your review and refuse to buy it! You decide to get more conniving and turn them into geometric diamonds! That is, you will choose four points on the boundary of the original lumps of coal and cut out the resulting quadrilaterals such that the new shapes have non-zero area, two pairs of parallel sides, and four equal length sides, also known as a diamond or rhombus.

The Problem:

Given the convex quadrilaterals defining the initial lumps of coal, find four points on the boundaries of each lump of coal that form a rhombus.

The Input:

The first input line contains a single integer, d ($1 \leq d \leq 10^5$), indicating the number of lumps of coal to turn into diamonds. Each lump of coal will be defined by four lines of input, each consisting of two integers, x and y ($|x|, |y| \leq 10^5$), denoting the coordinates of one of the corners of the coal.

It is guaranteed that the lump of coal is a non-degenerate convex quadrilateral. Each quadrilateral’s vertices will be given in counterclockwise order. No three adjacent points on a quadrilateral’s boundary will be collinear.

The Output:

For each lump of coal, output four lines, each with two real numbers, denoting the coordinates of the diamond you cut out. Output the diamond’s vertices in counterclockwise order (you may start with any vertex). Assume that there is a valid rhombus that can be cut for every input quadrilateral.

Your answer will be accepted if for all sets of points, each point lies within 10^{-6} of the original quadrilateral’s boundary, the shortest side length is within 10^{-6} of the longest side length, and the area of the rhombus is greater than 10^{-6} .

If there are multiple valid answers for a lump of coal, you may output any one of them, i.e., if an input quadrilateral can be cut in several correct diamonds, you may output any one of them.

Sample Input**Sample Output**

1 0 0 10 0 10 10 0 10	0.0 5.0 5.0 0.0 10.0 5.0 5.0 10.0
2 0 0 10 0 9 5 1 5 1 1 5 1 7 5 3 5	3.1415 0.0 8.1415 0.0 8.1415 5.0 3.1415 5.0 4.5311288759 5.0 2.2344355620 3.4688711241 3.4688711241 1.0 5.7655644380 2.5311288759

Note: In the first Sample Input, the quadrilateral happens to be a diamond, so outputting the same quadrilateral would have been correct as well.

UCF Local Contest (Final Round) — September 12, 2026

Sleeper Agents

filename: sleeper

Difficulty Level: Hard

Time Limit: 5 seconds

The Elephant Espionage Brigade has deployed several sleeper agents across the zoo to aid in their ongoing war against the zookeepers. Ellanah, one of the elite members of this brigade, has been tasked with developing the technology to activate the sleeper agents for various missions.

Being skilled with technology, Ellanah was able to hack into the zoo's mainframe and can display a message of one or more lowercase English letters on all screens within the zoo. She knows all sleeper agents have an activation code, which is a string composed of lowercase English letters, and if a sleeper agent sees a message which is a substring of their code, they will be activated.

At certain points in the war, Ellanah will have to send a single message to the screens of the zoo such that she activates a specific set of agents, without activating any agent not in the set. As her connection to the mainframe is sensitive, she wants to write the shortest possible message that activates this specific set of agents. She has now given you several queries of sets of agents to activate, and you must determine the minimum length of a message she must send to activate the agents in that set without activating any other sleeper agents, if possible.

The Problem:

Given the code for each sleeper agent and queries of sets of sleeper agents, for each set of agents, determine the minimum length message needed to activate only the agents in the set, if possible.

The Input:

The first input line contains two integers, n and q ($1 \leq n, q \leq 3 \times 10^5$), indicating (respectively) the number of agents and the number of queries.

Each of the next n input lines contains a string of lowercase English letters, providing the code for an agent (assume that the agents are numbered 1 through n). It is guaranteed that each code is at least one letter and that the sum of all string lengths is at most 3×10^5 .

The following $2q$ input lines will contain information on the queries, each query consisting of two input lines. The first line for each query will contain a single integer k ($1 \leq k \leq n$), corresponding to the number of agents in the queried set. The second input line for each query will then contain k integers between 1 and n (inclusive) indicating the agents to be activated. That is, if some value j is included in the set, that means agent j must be activated, i.e., the message must be a substring of the code for the j^{th} agent. It is guaranteed that all values given in the set will be distinct. It is also guaranteed that the sum of all k is at most 3×10^5 .

The Output:

For each query, output the minimum length of a message needed to activate only the sleeper agents in that queried set (note that the length must be at least 1). If it is not possible to activate only the agents in the queried set, output -1.

Sample Input

Sample Output

5 4	3
fox	-1
oxen	2
xebra	1
flea	
fowl	
1	
1	
2	
1 3	
2	
2 3	
3	
1 2 3	

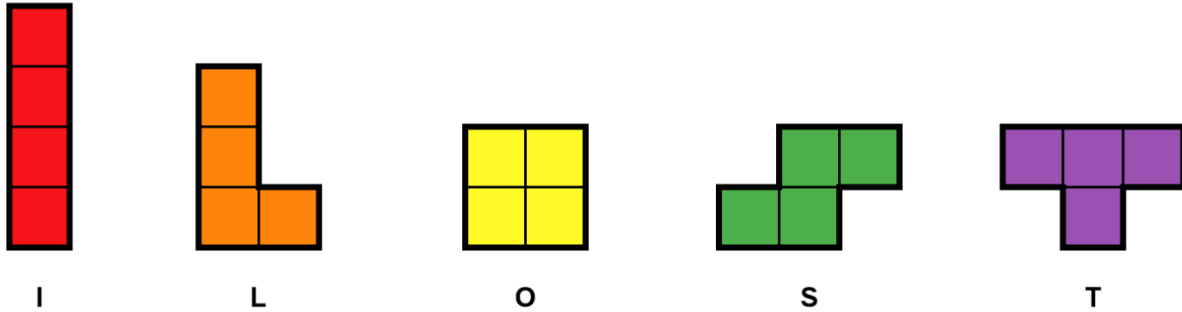
UCF Local Contest (Final Round) — September 12, 2026

Not another Polyomino!

filename: np

Difficulty Level: Hard

Time Limit: 1 second



Shown above are the five 4-cell polyominoes; formally, they are the five shapes that can be created by joining four unit squares edge to edge that are distinct under rotation and reflection. We can conveniently label each of these shapes with a single letter that looks vaguely similar to each polyomino. Listing them left to right, they are labeled I, L, O, S, T¹. Inspired by these shapes, the following packing puzzle has been designed:

Given the frequency of each piece type, we must fit them into the minimum possible number of 4×4 bins (i.e., a square with a side length of 4) with the following constraints:

- Each piece is placed into exactly one bin.
- Each piece is fully contained within the bin it is placed in (it can touch the sides).
- Pieces can touch each other on their boundaries, but no two pieces intersect with positive area.

Note that the polyominoes may be freely rotated, reflected, and translated inside the bins.

The Problem:

Given the frequency of each polyomino, determine the minimum number of 4×4 bins that are needed to pack away all of the polyominoes.

The Input:

The first input line contains a single integer, p ($1 \leq p \leq 100$), representing the number of packing puzzles you will be given; each puzzle is completely independent.

¹ You just lost the game.

The following p input lines provide the puzzles. Each puzzle is described by an input line containing five integers representing (respectively) the frequency of each piece in the order I, L, O, S, T ($0 \leq I, L, O, S, T \leq 10^6$).

The Output:

For each puzzle, print a line containing a single integer representing the minimum number of 4×4 bins needed to pack away the given polyominoes.

Sample Input

Sample Output

3	2
5 0 0 0 0	1
1 2 0 1 0	4
3 4 1 1 5	