University of Central Florida

Think Create Solve

2019 (Fall) "Practice" Local Programming Contest

Problems

Problem#	Difficulty Level	Filename	Problem Name
1	Easy	bulk	Buying in Bulk
2	Easy	stopping	Are We Stopping Again?
3	Medium	spell	Spell Checker
4	Medium	circle	Circles Inside a Square
5	Medium	divide	Anya's Favorite CD
6	Hard	sum	Sub Matrix Sum
7	Hard	mowing	Mowing Mischief

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

For example, if you are solving Anya's Favorite CD:

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

UCF “Practice” Local Contest — Aug 31, 2019

Buying in Bulk

filename: `bulk`
Difficulty Level: `Easy`
Time Limit: `3 seconds`

To encourage customers to shop more, some stores charge lower prices if you buy multiples of an item. For example, if you buy one, it may cost you \$5 but if you buy two, it will cost you \$8 instead of \$10.

The Problem:

Let’s assume a store provides discounts as follows:

1. No discount if you buy only one.
2. \$2 discount for each additional item if you buy more than one.

Given the number of items a customer has purchased and the price for one item, you are to compute the total cost for the customer.

The Input:

There is one input line, it provides two integers: the first integer c ($1 \le c \le 100$) is the number of items purchased by the customer, and the second integer p ($3 \le p \le 50$) is the price for one item.

The Output:

Print the total cost for the customer on a line by itself with no leading or trailing spaces.

Sample Input	**Sample Output**
`1 5`	`5`
`3 10`	`26`

Are We Stopping Again?

filename: `stopping`
Difficulty Level: `Easy`
Time Limit: `3 seconds`

Going on a road trip has been an adventure for Dr. Orooji and the programming team. Obviously he has to stop to refuel the car, but he also stops whenever the team members want to eat. Dr. O needs to figure out the number of stops before going on the trip so he is mentally prepared.

The Problem:

Find the total number of stops for Dr. O’s trip, given:

1. Total miles to be traveled.
2. How often he stops for gas (in miles).
3. How often he stops for food (in miles).

Assume that the car’s gas tank is full at the beginning of the trip and the team members are full as well. If the destination happens to be the time to refuel (or eat), do not count it as a stop. Also, if a particular mileage happens to be both refueling time and eating time, count it as one stop and not two stops.

Note that if a particular mileage happens to be refueling time only, the team members won’t eat at that stop. Similarly, if a particular mileage happens to be eating time only, the car is not refueled at that stop.

The Input:

There is one input line, it provides three integers (each between 1 and 1000, inclusive); these are the three values specified in order above.

The Output:

Print the number of stops for the trip on a line by itself.

Sample Input	**Sample Output**
`100 30 40`	`5`
`10 5 1`	`9`
`20 3 4`	`9`

UCF “Practice” Local Contest — Aug 31, 2019

Spell Checker

filename: `spell`
Difficulty Level: `Medium`
Time Limit: `3 seconds`

This question is about automatic spelling correction. Studies have shown that the majority of typing errors are caused by (let's assume the dictionary contains the word “these”):

1. Omitting one letter, e.g., the input word is “thse”.
2. Adding an extra letter, e.g., the input word is “thesce”.
3. Mistyping one letter, e.g., the input word is “thise”.
4. Transposing two adjacent letters, e.g., the input word is “tehse”.

With the aid of a dictionary, word processing systems can often detect and automatically correct these kinds of spelling errors.

The Problem:

You are to write a program that recognizes the four kinds of errors described above.

The Input:

The first input line contains an integer d ($1 \leq d \leq 100$) indicating the number of words in the dictionary. Each of the following d input lines contains a dictionary word. Assume these words start in column 1, contain at least 1 and at most 15 lowercase letters, and contain no other characters.

Following the dictionary words (i.e., the next input line), there is an integer n ($n \geq 1$) indicating the number of words to be spell checked. Each of the following n input lines contains a word. These words also start in column 1, contain at least 1 and at most 15 lowercase letters, and contain no other characters.

The Output:

Print each input word to be spell checked. Then, if the word is in the dictionary, print `CORRECT`. If the word is not in the dictionary, then find each word in the dictionary (in the order provided in the dictionary) for which the given input word might be a misspelling, and print the appropriate message from the following list:

`ONE LETTER OMITTED FROM` *word*
`ONE LETTER ADDED TO` *word*
`ONE LETTER DIFFERENT FROM` *word*
`TWO LETTERS TRANSPOSED IN` *word*

where *word* is a dictionary word. If the input word is not CORRECT and none of the above messages apply, then print UNKNOWN.

Note that two or more of the above messages might be applicable to an input word, and that one message might apply for more than one dictionary word. Note, however, that for a given input word and given dictionary word, at most one of the above messages apply. For each input word, you are to process the dictionary words in the order provided in the input and print all messages that are valid.

Leave a blank line after the output for each input word. Follow the format illustrated in Sample Output.

Sample Input

```
7
ali
thru
tu
funfunfun
the
tuh
th
3
ali
thu
orooji
```

Sample Output

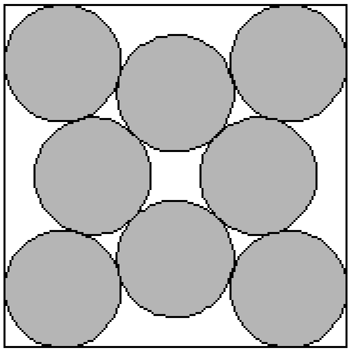
```
ali
CORRECT

thu
ONE LETTER OMITTED FROM thru
ONE LETTER ADDED TO tu
ONE LETTER DIFFERENT FROM the
TWO LETTERS TRANSPOSED IN tuh
ONE LETTER ADDED TO th

orooji
UNKNOWN

```

UCF "Practice" Local Contest — Aug 31, 2019

Circles Inside a Square

filename: `circle`
Difficulty Level: `Medium`
Time Limit: `3 seconds`

You have 8 circles of equal size and you want to pack them inside a square. You want to minimize the size of the square. The following figure illustrates the minimum way of packing 8 circles inside a square:



The Problem:

Given the radius, *r*, find the area of the minimum square into which 8 circles of that radius can be packed.

The Input:

There is one input line, it consists of a positive real number (between 0.001 and 1000, inclusive) denoting the radius, *r*.

The Output:

Print the area of the minimum square where 8 circles of radius *r* can be packed. Print 5 digits after the decimal. Your output is considered correct if it is within ±0.00001 of the judge's output.

Sample Input	**Sample Output**
`0.1`	`0.34383`
`0.2`	`1.37532`

Anya’s Favorite CD

filename: `divide`
Difficulty Level: `Medium`
Time Limit: `3 seconds`

A few years ago, Arup started submitting problems to the UCF Local Contest regarding her daughter Anya’s CD requests. Unfortunately, his latest question wasn’t good enough to make the cut for the 2019 UCF Local Contest. Luckily, a couple coaches, nostalgic for solving Anya’s CD problems, suggested that Arup’s latest question about Anya’s CD requests be added to the practice local contest. Arup is so ecstatic that his question is being used in this contest that he will personally present the first solver with a copy of Ed Sheeran’s CD “Divide”.

To this day, Arup drives Anya to school daily and Anya makes CD requests in the car. When Anya was four (in 2017), she would request a sequence of track numbers for Arup to play and Arup had to determine the fewest number of button presses to satisfy the requests. If you didn't compete in 2017, here is an explanation of how the CD player in Arup's car works:

Arup can only change tracks by pressing a forward button or a backward button. If track number k has completed, when Arup presses the backward button, track number k will play again. Alternatively, if he presses the forward button when track k has completed, then track k+2 will play. The only exception to the latter is that if track k+2 doesn't exist. In this case, the tracks just wrap back around to the beginning, starting with 1, then 2, etc. Similarly, if Arup presses the backward button twice right after track 1 completes, this will move the CD to track t, where t is the number of tracks on the CD. In the absence of a button press, after track k completes, track k+1 plays, except for when k = t, in which case, track 1 plays next.

Now that Anya is six, her requests have become slightly more flexible. Instead of insisting on particular songs in a sequence, for each song, Anya gives Arup several choices[1]. For example, for the first song, Anya may tell Arup that she wants to hear one of tracks 1, 3, 8 or 9, and for the second song Anya may tell Arup she wants to hear one of tracks 2 or 12.

Even though Arup has some choice in what songs he plays, he still gets caught pressing either the forward or backward button a great deal. Help him minimize the number of times he presses the buttons. In the example above, given that the CD is originally queued to start with track 1, no button presses are required because he can simply select to play track 1 followed by track 2.

The Problem:

Given the number of tracks on Anya's favorite CD and the set of possible tracks for each song Anya wants played, determine the minimum number of button presses Arup must make to get a valid sequence of songs played. For the purposes of this problem, assume that at the very

[1] Currently, Anya’s favorite CD is Divide by Ed Sheeran, and her favorite track numbers on this CD are 4, 5, and 7.

beginning the CD player is queued up to play track number 1, so that if the first song played is anything but track 1, some buttons will have to be pressed before the first song plays.

The Input:

The first input line consists of two (space separated) positive integers: t ($1 \le t \le 10^9$), the number of tracks on Anya's favorite CD and *s* ($1 \le s \le 1000$), the number of songs Anya would like to listen to from the CD. The next *s* input lines contain the possible track numbers for each song she would like to listen to. The i^{th} of these lines starts with a positive integer, n_i ($1 \le n_i \le 10$), representing the number of choices Anya has provided for the i^{th} song she listens to. This is followed by n_i distinct positive integers, each in between 1 and *t*, inclusive, indicating the track numbers of the possible i^{th} song Anya will listen to.

The Output:

Output a single integer on a line by itself indicating the minimum number of button presses Arup can use to play the desired sequence of songs from the CD.

Sample Input	**Sample Output**

`15 2`
`4 1 3 8 9`
`2 12 2`
	`0`

`12 5`
`2 5 7`
`2 5 7`
`3 12 2 4`
`1 9`
`2 4 5`
	`16`

UCF “Practice” Local Contest — Aug 31, 2019

Sub Matrix Sum

filename: `sum`
Difficulty Level: `Hard`
Time Limit: `3 seconds`

You have written many programs to search mazes so matrix search shouldn’t be any different, or will it?

The Problem:

An integer matrix with R rows and C columns has $\binom{R}{2}\binom{C}{2}$ sub matrices. We want to select a sub matrix with sum (the sum of all integers in it) greater than or equal to a given integer S. We want the size of the sub matrix to be the least possible. The size of a sub matrix is defined as the number of elements in that sub matrix (i.e., number of rows * number of columns in that sub matrix).

The Input:

The first input line consists of three integers *R*, *C* ($1 \le$ R $\le$ 100,000; $1 \le$ C $\le$ 100,000; $1 \le$ R*C $\le$ 100,000) and *S*. Next R lines contain the description of the matrix. Each of these R lines contains C integers separated by a single space. All integers (other than R and C) are between -10^9 and $+10^9$, inclusive.

The Output:

Print the size of the minimum sub matrix whose sum is greater or equal to the given S. If there is no such sub matrix, output -1.

Sample Input	**Sample Output**

`3 3 26`
`1 2 3`
`4 5 6`
`7 8 9`
	`4`

`3 3 0`
`-1 -2 -3`
`-4 -5 -6`
`-7 -8 -9`
	`-1`

`2 2 1`
`-1 -2`
`0 2`
	`1`

Mowing Mischief

filename: `mowing`
Difficulty Level: `Hard`
Time Limit: `3 seconds`

Bessie's younger cousins, Ella and Bella, are visiting the farm. Unfortunately, they have been causing nothing but mischief since they arrived.

In their latest scheme, they have decided to mow as much grass as they can. The farm's prime grassland is in the shape of a large $t \times t$ square. The bottom-left corner is (0, 0), and the top-right corner is (t, t). The square therefore contains $(t+1)^2$ lattice points (points with integer coordinates).

Ella and Bella plan to both start at (0, 0) and run at unit speed to (t, t) while each holding one end of a very sharp and very stretchy wire. Grass in any area that is swept by this wire will be cut. Ella and Bella may take different paths, but each path consists of only upward and rightward steps, moving from lattice point to lattice point.

Bessie is rather concerned that too much grass will be cut, so she invents a clever plan to constrain the paths Ella and Bella take. There are n yummy flowers scattered throughout the grassland, each on a distinct lattice point. Bessie will pick a set of S flowers that will be required for both Ella and Bella to visit (so Ella's path must visit all the flowers in S, and so must Bella's path). In order to add as many waypoints to these paths as possible, Bessie will choose S to be as large as possible among subsets of flowers that can be visited by a cow moving upward and rightward from (0, 0) to (t, t).

The Problem:

Ella and Bella will try to minimize the amount of grass they cut, subject to the restriction of visiting flowers in S. Please help Bessie choose S so that the amount of grass cut is as small as possible.

The Input:

The first input line contains two positive integers: n ($1 \le n \le 2(10)^5$), indicating the number of yummy flowers scattered throughout the grassland and t ($1 \le t \le 10^6$), where (t, t) are the coordinates of the top right corner of the grid.

Each of the next n lines contains the integer coordinates (x_i, y_i) of a flower, with $1 \le x_i, y_i \le t$-1, for all i ($1 \le i \le n$), and no two flowers lie on the same horizontal or vertical line.

The Output:

Output a single integer on a line by itself giving the minimum possible amount of cut grass.

Sample Input	Sample Output

5 20
19 1
2 6
9 15
10 3
13 11
	117

Sample Explanation:

In this sample, it's best to pick the flowers at (10, 3) and (13, 11). With this selection, the first rectangle of grass cut has dimensions 10 x 3, the second one has 3 x 8, and the last one has the dimensions 7 x 9. The total area of these rectangles is 30 + 24 + 63 = 117. A sub-optimal choice would be taking the flowers at (2, 6) and (9, 15). This choice cuts a rectangle of size 2 x 6, a second rectangle of size 7 x 9 and a third rectangle of size 11 x 5 for a total area of 12 + 63 + 55 = 130, which is greater than 117. The third possible choice which is sub-optimal would be to pick the flower at (2, 6) and the flower at (13, 11), which results in a total area of grass cut of 152 units^2.

Problem Credits: USA Computing Olympiad – Dhruv Rohatgi