University of Central Florida

2022 Local Programming Contest (Qualifying Round)

Problems

Problem#	Difficulty Level	Filename	Problem Name
1	Easy	easyexpr	Easy-to-Solve Expressions
2	Easy	easyword	Easy-to-Pronounce Words
3	Easy	dexor	Decimal XOR
4	Easy-Medium	fish	Square Fishing Net
5	Medium	aqualin	Aqualin
6	Medium	rcv	RCV Simplification
7	Medium	maze	Panda Hunting Treasure Box
8	Medium-Hard	lecture	Lecture Allocation
9	Medium-Hard	smooth	Stone Smoothing

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

For example, if you are solving Decimal XOR:

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

UCF Local Contest (Qualifying Round) — September 3, 2022

Easy-to-Solve Expressions

filename: `easyexpr`
Difficulty Level: `Easy`
Time Limit: `5 seconds`

When one looks at a set of numbers, one usually wonders if there is a relationship among them? This task is more manageable if there are only three numbers.

The Problem:

Given three distinct positive integers, you are to determine how one can be computed using the other two. Print `1` if any of the three numbers is the *sum* of the other two numbers, print `2` if any of the three numbers is the *product* of the other two numbers, print `3` otherwise. Assume that exactly one of these three messages will apply.

The Input:

There is only one input line; it contains three distinct positive integers, each between 2 and 1000 (inclusive).

The Output:

Print the appropriate message as described above.

Sample Input	Sample Output
`10 30 20`	`1`
`10 20 200`	`2`
`100 5 700`	`3`

Easy-to-Pronounce Words

filename: `easyword`
Difficulty Level: `Easy`
Time Limit: `5 seconds`

We define a word as *easy-to-pronounce* if every vowel in the word is immediately followed by a consonant and every consonant in the word is immediately followed by a vowel. The first letter of the word can be a vowel or consonant. Assume that the vowels are: `a, e, i, o, u` (note that the letter `y` is not a vowel, i.e., it is a consonant).

The Problem:

Given a word, print `1` (one) if it is easy-to-pronounce, `0` (zero) otherwise.

The Input:

There is only one input line; it contains a word consisting of 1-30 lowercase letters (starting in column 1). Assume that there will not be any characters other than the lowercase letters in the input.

The Output:

Print the appropriate message as described above.

Sample Input	**Sample Output**
contest	0
coaches	0
cocahes	1
ali	1

Decimal XOR

filename: `dexor`
Difficulty Level: `Easy`
Time Limit: `5 seconds`

The binary operation XOR accepts two binary digits as input and outputs a binary digit: if both input digits are `0` (or both are `1`), the output is `0`; otherwise the output is `1`. We can look at this as: if both input values are `low` (or both are `high`), the output is `0`; otherwise the output is `1`.

Decimal numbers have several digits and each digit can be one of 10 values (0-9). We define the operation DEXOR (XOR of two decimal numbers) as follows: we DEXOR two decimal digits at a time; the two decimal digits at 1^{st} position are DEXOR'ed, the two decimal digits at 10^{th} position are DEXOR'ed, the two digits at 100^{th} position are DEXOR'ed, etc. When DEXOR'ing two decimal digits, the result digit is `0` if both digits are too small ($\leq$ `2`) or both digits are too large ($\geq$ `7`); the result digit is `9` otherwise.

The Problem:

Given two decimal numbers, compute their DEXOR.

The Input:

There are two input lines, each line providing a decimal number between 0 and 999,999 (inclusive). Assume that there will not be extra leading zeroes in an input number, i.e., there will not be extra zeroes at the beginning of a number in the input.

The Output:

Print the DEXOR of the two decimal numbers. When DEXOR'ing two decimal numbers, if one has fewer digits, it should be considered as having zeros on the left to make both numbers having the same number of digits. The result should have as many digits as the larger number.

Sample Input	**Sample Output**
`22776` `15954`	`09099`
`29` `18908`	`09900`

Note that, in the second Sample, `29` should be treated as `00029` so that it will have the same number of digits as the second number (so that they can be DEXOR'ed digit-by-digit).

Square Fishing Net

filename: `fish`
Difficulty Level: `Easy-Medium`
Time Limit: `5 seconds`

With so many activities/events being virtual these days, we are going on a virtual fishing trip!

The Problem:

Given the (x,y) coordinates of n points (each point represents a fish) and a square (representing a fishing net), what is the maximum fish you can catch with one try? You can place the square net anywhere but its sides must be parallel to X-axis and Y-axis. A fish is caught if it is inside or on the boundary of the net.

The Input:

The first input line contains two integers: s $(1 \le s \le 100)$, indicating the length of one side of the fishing net and n $(1 \le n \le 100)$, indicating the number of fish. Each of the next n input lines contains two integers (each between 1 and 100, inclusive) indicating the (x,y) coordinates of one fish. Assume that no two fish are at the same location.

The Output:

Print the maximum number of fish you can catch.

Sample Input	**Sample Output**
3 8 2 1 2 3 5 1 5 2 3 2 4 2 10 5 11 5	6
50 2 10 5 11 5	2

Aqualin

filename: `aqualin`
Difficulty Level: `Medium`
Time Limit: `5 seconds`

The board game Aqualin is played on an $n \times n$ grid, where each grid cell is filled with a piece. Each piece is an animal of a particular color, i.e., each piece represents two properties: animal type and animal color. For simplicity, we assign letters 'A' through 'Z' for animal types and number the different colors 1 to n.

After each of the n^2 cells of the grid are filled, the game is scored. There are two teams.

The first team gets points for the largest connected component of animals of the same type that are size 2 or greater. Specifically, for each of the largest connected components of the same type of animals, the first player gets $1 + 2 + \ldots + (c - 1)$ points, where c is the number of animals in the component. For example, if there are connected components of 3 starfish ('A'), 4 octopi ('B'), 1 whale ('C'), and 2 starfish ('A'), then points would be awarded for only the first two groups of animals. **Note that no score is awarded to a component of size 1 and, when there are several connected components of the same animal type, only the largest connected component is awarded points.** Thus, in the case discussed above, the first team would get 1 + 2 = 3 points for the starfish, and 1 + 2 + 3 = 6 points for the octopi, for a total of 9 points. A connected component of animals is the set of animals you can reach directly or indirectly by traveling up, down, left or right in the grid by only going through animals of the same type.

The second team gets points for the largest connected component of animals of the same color. The scoring is the same as previously described, based on component size.

Here is an example 5 × 5 grid, filled out. In each cell, an ordered pair (x, y) indicates that the animal in that cell is of type x, color y.

(B,3)	(A,1)	(C,1)	(A,2)	(A,5)
(B,4)	(B,1)	(B,5)	(E,4)	(E,3)
(C,3)	(C,2)	(B,2)	(D,2)	(E,2)
(A,3)	(C,4)	(A,4)	(E,5)	(D,1)
(D,3)	(C,5)	(D,4)	(D,5)	(E,1)

First, let's consider the score by animal type. There are two animals of type 'A' in the top right corner for a score of 1. All five animals of type 'B' are connected (look at top left) for a score of 10. Four animals of type 'C' are connected, starting from the animal (C, 3) for a score of 6. Two animals of type 'D' are connected, (D, 4) and (D, 5), for a score of 1. Three animals of type 'E' are connected, (E, 4), (E, 3) and (E, 2), for a score of 3. The total score for the first team is 1 + 10 + 6 + 1 + 3 = 21.

There are three animals of color 1 connected at the top for a score of 3. Note that there are 2 other animals of color 1 connected in the bottom right corner, but this score doesn't count because we only count the largest connected component of a single type (or color) of an animal. There are 4 animals of color 2 connected (all on row 3) for a score of 6. There are 3 animals of color 3 connected (all on column 1) for a score of 3. There are 3 animals of color 4 connected, (C, 4), (A, 4) and (D, 4), for a score of 3. There are 2 animals of color 5 connected, (E, 5) and (D, 5), for a score of 1. The total score for the second team is 3 + 6 + 3 + 3 + 1 = 16.

Note: In the given example, there are five animal types and five animal colors. Although each of the 25 possible animal type/color combinations appeared exactly once in this example, this is not guaranteed to be true for all input grids. That is, some animal type/color combinations may appear more than once and some animal type/color combinations may not appear at all.

The Problem:

Given the contents of the grid at the end of a game, determine the score for both teams.

The Input:

The first input line contains a single integer: n ($2 \le n \le 26$), indicating the number of rows and columns in the game grid. Each of the following n input lines provides the contents of one row in the grid. Each row is represented by n terms, each term providing the type x ('A' $\le x \le$ 'Z') and color y ($1 \le y \le n$) of the corresponding animal on the grid. Assume that the input rows start in column one and there is exactly one space separating different values on these input lines.

The Output:

Print, on a line by itself, the score for the first team, followed by a space, followed by the score for the second team.

Sample Input	Sample Output
5 B 3 A 1 C 1 A 2 A 5 B 4 B 1 B 5 E 4 E 3 C 3 C 2 B 2 D 2 E 2 A 3 C 4 A 4 E 5 D 1 D 3 C 5 D 4 D 5 E 1	21 16
2 A 1 B 1 A 1 B 1	2 6

UCF Local Contest (Qualifying Round) — September 3, 2022

RCV Simplification

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

The following is from Ballotpedia [https://ballotpedia.org/Ranked-choice_voting_(RCV)]:

Broadly speaking, the *ranked-choice voting process* unfolds as follows for single-winner elections:

1. Voters rank the candidates for a given office by preference on their ballots.
2. If a candidate wins an outright majority of first-preference votes (i.e., 50 percent plus one), he or she will be declared the winner.
3. If, on the other hand, no candidates win an outright majority of first-preference votes, the candidate with the fewest first-preference votes is eliminated.
4. All first-preference votes for the failed candidate are eliminated, lifting the second-preference choices indicated on those ballots.
5. A new tally is conducted to determine whether any candidate has won an outright majority of the adjusted voters.
6. The process is repeated until a candidate wins a majority of votes cast.

Example: Assume that there are four candidates in an election. The table below presents the raw first-preference vote totals for each candidate:

Raw first-preference vote tallies		
Candidate	**First-Preference Votes**	**Percentage**
Candidate A	475	46.34%
Candidate B	300	29.27%
Candidate C	175	17.07%
Candidate D	75	7.32%

In the above scenario, no candidate won an outright majority of first-preference votes. As a result, the candidate (Candidate D) with the smallest number of first-preference votes is eliminated. The ballots that listed candidate D as the first preference are adjusted, raising their second-preference candidates. Assume that, of the 75 first-preference votes for Candidate D, 50 listed Candidate A as their second preference and 25 listed Candidate B. The adjusted vote totals would be as follows:

Adjusted vote tallies		
Candidate	**Adjusted First-Preference Votes**	**Percentage**
Candidate A	525	51.22%
Candidate B	325	31.71%
Candidate C	175	17.07%

On the second tally, Candidate A secured 51.22 percent of the vote, thereby winning the election.

Note:

If several candidates are tied for the fewest first-preference votes, all such candidates are eliminated. So, candidates not eliminated must have at least one more first-preference vote than those eliminated.

The Problem:

We have received information on the percentage for the first-preference for each candidate, but we don't know how the candidates are listed as the second preference, third preference, etc. Help write a program to remove candidates that cannot possibly win. More specifically, given the current votes for a set of candidates, find the set of candidates that cannot possibly win.

The Input:

The first input line contains a single integer, N ($1 \le N \le 100{,}000$), representing the number of votes. Each of the following N input lines contains a candidate name receiving the first-place vote from that voter. Each candidate name is 1-25 letters (lowercase and uppercase), starting in column one.

The Output:

On the first output line, print a single positive integer, C, the number of candidates that cannot win. Each of the remaining C output lines should contain a candidate name that cannot win. The candidate names should be printed in lexicographical order (increasing order).

Sample Input	**Sample Output**
5 Alice Alice Bob Bob Carol	1 Carol
2 Alice Bob	2 Alice Bob

Panda Hunting Treasure Box

filename: `maze`
Difficulty Level: `Medium`
Time Limit: `2 seconds`

Our favorite treasure hunter, Panda, has been dropped in a two-dimensional maze with several treasure boxes. Each maze cell is either empty or has one treasure box in it. The treasure boxes contain money and no two boxes have the same amount, i.e., they are all distinct amounts. Panda can have only one of these boxes and, obviously, he would prefer the one with the highest amount but he may not have enough energy to get to the cell containing that box so Panda may have to settle for a lower amount.

The Problem:

Panda starts with some energy. The picture to the right shows the energy needed for Panda to move into any of the eight neighboring cells (note that the boundary cells in the maze have fewer than eight neighbors).
To illustrate the energy needed for movements, if Panda wants to move up, it requires 2 units of energy; if he wants to move down, it requires 6 units of energy.

1	2	3
8	Panda	4
7	6	5

If Panda moves into a cell with a treasure box, that box is what he gets, i.e., the journey is over even if Panda still has some energy left to make more moves. And, obviously, Panda can make a move only if he has enough energy for that move. When Panda makes a move, his energy will go down accordingly to that move.

You are to determine the highest amount Panda can get. Note that Panda does not have to use all his energy, i.e., it is ok if Panda still has some energy left when he gets a treasure box.

The Input:

The first input line contains five integers:

- R_m ($2 \le R_m \le 500$), indicating the number of rows in the maze;
- C_m ($2 \le C_m \le 500$), indicating the number of columns in the maze;
- R_p ($1 \le R_p \le R_m$), indicating the row number of Panda's starting position;
- C_p ($1 \le C_p \le C_m$), indicating the column number of Panda's starting position; and
- E_p ($1 \le E_p \le 10^6$), indicating Panda's starting energy.

Assume that Panda's starting position is valid (i.e., it is in the maze) and that cell does not contain a treasure box.

Each of the next R_m input lines contains C_m integers (each integer between 0 and 10^6, inclusive). Each input line describes one row in the maze, providing the cell contents. A zero indicates there is no treasure box in that cell; non-zero indicates a treasure box (the amount in the box). Assume that there is at least one treasure box in the maze and that Panda has enough starting energy to reach one treasure box.

The Output:

Print the highest amount Panda can get.

Sample Input	**Sample Output**
4 3 2 1 15 0 0 0 0 0 0 0 0 10 0 20 0	10
4 3 2 3 50 0 0 0 0 0 0 0 0 10 0 20 0	20
4 2 1 1 90 0 0 0 0 10 30 40 50	30

Lecture Allocation

filename: `lecture`
Difficulty Level: `Medium-Hard`
Time Limit: `2 seconds`

You are the coordinator for a competitive programming club. You need to hire some teachers to give lectures. There are a fixed number of lectures that need to be given this year. Additionally, there are a limited number of teachers that are willing to give lectures. Each teacher can teach up to three lectures, but not all the teachers need to teach a lecture, i.e., a teacher could teach 0, 1, 2, or 3 lectures. Each teacher charges a different amount depending on the number of lectures they give.

The money not spent will be used to fly the team to other contests, so you want to spend as little money as possible hiring enough teachers to give all the lectures.

The Problem:

Given the number of lectures to teach and how much each teacher charges for giving the lectures, determine the least amount of money necessary such that all the lectures will be taught.

The Input:

The first input line contains two integers, L and T ($1 \leq L \leq 5000$, $L/3 \leq T \leq L$), representing (respectively) the number of lectures and the number of teachers. Each of the following T input lines contains three integers, the i^{th} of which contains a_{i1}, a_{i2}, and a_{i3} ($0 < a_{i1} < a_{i2} < a_{i3} \leq 100{,}000$), representing (respectively) how much the i^{th} teacher charges to give 1, 2, and 3 lectures.

The Output:

Print on a single line by itself a single positive integer: the least cost for paying the teachers to cover all L lectures. Assume that there are enough teachers to cover all the lectures.

Sample Input **Sample Output**

Sample Input	Sample Output
4 3 8 10 20 10 20 30 11 17 25	27
6 2 10 20 25 30 35 37	62
5 2 10 20 25 30 35 37	57

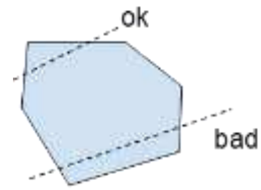
Notes:

For the first Sample Input, the first teacher can give two lectures and the third teacher can give two lectures, so the total cost is `10 + 17 = 27`.

For the third Sample Input, the first teacher can give two lectures and the second teacher can give three lectures, so the total cost is `20 + 37 = 57`.

Stone Smoothing

filename: `smooth`
Difficulty Level: `Medium-Hard`
Time Limit: `1 second`



You found a really sharp stone that could be described as a convex polygon. You decided to use some skills to make it smoother. You can "smooth" a corner (vertex) of the stone (polygon) by shaving it down some amount as long as you don't smooth down another corner at the same time, i.e., only one corner can be smoothed at a time.

Note that, when one vertex (corner) is smoothed, it is replaced by two new vertices. So, if desired, these new vertices (corners) can be smoothed.

To make the smoothest stone, you have to ensure that the largest exterior angle of the polygon is as small as possible, after smoothing a given stone a fixed number of times.

Geometry Refresher: An **exterior angle of a polygon** is an angle at a vertex of the polygon, outside the polygon, formed by one side and the extension of an adjacent side. In other words, if the side of a polygon is extended, **the angle formed outside the polygon** is the exterior angle.

The Problem:

Given the shape of the stone described using *xy*-coordinates and the number of times the stone could be smoothed, determine the smallest possible value for the largest resulting exterior angle.

The Input:

The first input line contains two integers, C and S ($1 \le C \le 5{,}000$; $1 \le S \le 10^9$), representing (respectively) the number of corners in the stone and the number of times the stone can be smoothed. Each of the following C input lines contains two integers, the ith of which are x_i and y_i ($-100{,}000 \le x_i, y_i \le 100{,}000$), representing the x coordinate and the y coordinate of the ith corner of the stone (the vertices of the polygon are given in clockwise order).

The Output:

Print on a single line by itself a single floating-point number: the minimum possible value for the largest exterior angle after performing all the smoothing. Answers within 10^{-6} absolute or relative of the expected answers will be considered correct.

Sample Input	**Sample Output**
3 6 0 0 0 10 10 0	225.00000000000003
4 3 -1 1 1 1 1 -1 -1 -1	270.00000000000006