Thirteenth Annual
University of Central Florida

High School Programming Tournament: Online Edition

Problems – Division 2

Problem Name	Filename
Combine Me!	combine
Mutual Friends	mutual
Passing Notes	notes
Rhino Rampage	rhino
Movie Sequel	sequel
Suspicious Shapeshifter	shapeshifter
Skipping Stones	stones
Sugar Honeycomb	sugar
Zigzag Palindrome	zigzag

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

For example, if you are solving Zigzag Palindrome,
Call your program file:
zigzag.cpp, zigzag.java, or zigzag.py
Call your Java class: zigzag

Combine Me!

Filename: `combine`

Ryan is playing a Battle Royale word game. In this game, you are given a list of words, and you must pick two different words and concatenate them together. Your score is equal to the number of *distinct* characters in the combined word.

The Problem:

Ryan would like to know for the given list of words, what two words he should pick to maximize his score?

The Input:

The first line of input will contain a single, positive integer, t, representing the number of rounds Ryan played. For each round, several lines will follow. The first line for each round will contain a single integer, n ($2 \leq n \leq 100$), representing the number of words provided in the round. The following n lines each contain a string, s (of length between 1 and 100, inclusive), of lowercase letters, representing each of the words.

The Output:

For each round, output a single line containing the two chosen words separated by a single space. If there are multiple correct answers, any of them will be accepted.

Sample Input:

```
2
8
buy
freeze
squeeze
roll
juice
combine
pill
sell
3
abc
a
b
```

Sample Output:

```
squeeze combine
abc a
```

Mutual Friends

Filename: `mutual`

You have been hired to work at Facebook, which is a company name that is completely original and fictional because there is no company that is named like that right now. Your first task is to implement a mutual friends algorithm. There are n accounts on the website. You need to find the mutual friends between the CEO (account #1) and you (account #n).

A mutual friend between you and the CEO is any account, which is friends with you and the CEO.

The Problem:

Given a list of friendship relations between two accounts and the total number of accounts on the website, find the mutual friends between your account and the CEO.

The Input:

The first line of input will be a single, positive integer, t, representing the number of test cases. Each test case begins with two integers, n and m ($2 \leq n \leq 100$; $1 \leq m \leq 200$), representing the number of accounts and the number of friend relationships on the website, respectively. Then m lines follow, each containing a pair of integers, u and v ($1 \leq u \leq n$; $1 \leq v \leq n$), denoting that account u and v are friends. It is guaranteed that no account will be friends with itself, and no friendship pairs will show up multiple times within a test case. For example, if a test case contains the pair "1 2", it will only be provided once, and "2 1" will not be included (as it is redundant information).

The Output:

For each test case, first output on a line by itself a single integer, k, denoting the number of mutual friends. Then, follow this with k lines, each containing the account ID of the mutual friends. The account IDs can be given in any order. Output a blank line after the output for each test case.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
3
10 10
1 10
1 2
1 3
1 4
1 5
10 2
10 3
10 4
10 6
5 6
2 1
1 2
5 7
1 2
1 3
2 3
2 4
1 4
5 4
5 2
```

Sample Output:

```
3
2
3
4

0

2
2
4
```

Passing Notes

Filename: `notes`

O.M.G! Can you believe what Jordan said to Alex at lunch today?! I totally can't wait to tell my B.F.F.O.U.C. (Best Friend Forever, Or Until College) in math class today! The thing is, though, Mr. Davis is tooootally a drag, and like, he doesn't let us talk in class! That's, like, legit lame. Anyway, I'm gonna pass a note to my bestie, but some of those dweebs along the way might get caught! If I tell you how likely each one of them is to get caught, could you, like, tell me the chances my note gets through? That would totally rock!

The Problem:

Given the deets on the classmates between me and my B.F.F.O.U.C., tell me the chances my note gets through! Uh, I mean, please.

The Input:

The first line of the input will contain a single, positive integer, t, representing the number of notes. Each note will be defined across two lines. The first line of each note will contain a single integer, n ($0 < n < 10$), representing the number of classmates to pass notes. The next line will contain n integers from 0 to 100, the percentage chance that each classmate will get caught.

The Output:

For each note, output a line containing a single integer, the percentage chance the note gets through every classmate. Round down to the nearest percentage point.

Sample Input:

```
3
3
50 10 10
4
2 0 2 2
3
5 100 5
```

Sample Output:

```
40
94
0
```

Rhino Rampage

Filename: `rhino`

In the auto-battler Super Auto Pets, players select a team of animals in a row, and then the teams clash, dealing damage to one another until one animal remains. The rules of the game are as follows. First, each animal has an attack value and a health value. Then, the animals at the front of the lines both simultaneously attack each other, each dealing its own attack to the other, removing that value from the opponent animal's health. If one or both of the animals' health drops to zero (or below zero), the animal is Knocked Out, and the next animal behind it in line comes up to take its place. A player wins once they have a surviving animal but their opponent does not.

The Rhino has a special ability that when it knocks out an opponent (and survives) it deals 5 damage to the new enemy in the front. This will chain, so if the rhino knocks out an opponent, it will deal 5 damage to the new enemy in front, and if that enemy is knocked out, it will deal 5 damage to the next, etc. until the entire enemy team is wiped out, or one of the enemies survives the hit. You and your friend both love rhinos, and have a team with only rhinos. The problem is, the game animations can take a long time to run, so you want to write a program to determine which team will win.

The Problem:

Given the stats of your team and your opponent's team, determine the winner of the all-rhino matchup.

The Input:

The first line of the input will contain a single, positive integer, g, representing the number of games you and your friend played. The first line of each game will contain two integers, n and m ($1 \le n \le 1{,}000$; $1 \le m \le 1{,}000$), representing the number of rhinos on your team and the number of rhinos on your opponent's team, respectively. The next n lines will contain two integers, a_i and h_i ($1 \le a_i \le 1{,}000$; $1 \le h_i \le 1{,}000$), the attack and health, respectively, of your i^{th} rhino from the front to the back of the line, respectively. Then, m lines will follow, each line containing two integers in the same format, denoting the attack and health of your opponent's rhinos.

The Output:

For each game, output a line containing either "`I win!`" or "`You win!`" depending on who won. If both players lose their last animal on the same turn, output "`Draw!`" instead.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
2
3 2
1 4
2 3
5 1
1 6
1 1
1 3
4 5
3 3
18 1
2 9
```

Sample Output:

```
You win!
Draw!
```

Movie Sequel

Filename: `sequel`

It's a common trope that movie sequels are never as good as the original, but their names can be quite compelling. The production company knows that fans love to see their favorite characters return to the screen, so they've decided to stick to a simple naming scheme for their sequels. They need you to write a program to generate these sequel names.

The Problem:

Given the title of the original movie, generate the title of the sequel using the following format:

<*Original Title*> 2: Return of The <*Original Title*>!

The output must match this naming scheme exactly.

The Input:

The first line of input contains a single, positive integer, *t*, representing the number of movies. Each movie contains a single string, the name of the original movie. The string may contain any letters or punctuation, but it will not contain any spaces. Its length will be between 1 and 100 characters (inclusive).

The Output:

For each movie, output a single line containing the name of the sequel using the specified format.

Sample Input:

```
3
Octagon
X-Ray
Knights
```

Sample Output:

```
Octagon 2: Return of The Octagon!
X-Ray 2: Return of The X-Ray!
Knights 2: Return of the Knights!
```

Suscipious Shapeshifter

Filename: `shapeshifter`

There is a shapeshifter among us! This shapeshifter is able to take any form as long as they maintain the same area (we all live in a 2D plane after all!). Crew members can be modeled by circles, and we absolutely despise anything square. Recently several of the crew members have been found in square-shock, a fate worse than death! In order to figure out who our shapeshifter must be, we need to know what the shapeshifter's side length must be if they shapeshifted into a square. Given a bunch of crew member's radii, can you help us determine what the corresponding side length of the square must be?

The Problem:

Given the radius of a crew member, determine what the side length of their square form must be assuming that area is conserved.

The Input:

The first line will consist of a single integer, ***c***, the number of crew members to investigate. The following ***c*** lines will consist of a single integer, r ($1 \leq r \leq 10^6$), the radius of the crew member.

The Output:

Print ***c*** lines each consisting of the side length of the shapeshifting square. Answers within absolute or relative error of 10^{-6} of the judge answer will be accepted.

Sample Input:

```
3
1
5
12
```

Sample Output:

```
1.772453851
8.862269255
21.269446211
```

Skipping Stones

Filename: `stones`

Alice and Bob have decided to skip stones at the lake on a beautiful sunny day. Alice gathered n piles of stones and Bob gathered m piles of stones. However, before they start skipping stones, Alice is afraid that Bob might have a greater total amount of stones and therefore will have more fun than her.

The Problem:

Given Alice has n piles of stones where a_i ($1 \leq i \leq n$) represents that there are 2^p stones (where $p = a_i$) in each of Alice's pile and given Bob has m piles of stones where b_j ($1 \leq j \leq m$) represents that there are 2^q stones (where $q = b_j$) in each of Bob's pile, find out who has more stones in total.

The Input:

The first line of the input will contain a single, positive integer, d, representing the number of days Alice and Bob have decided to skip stones. For each day d there will be 4 lines of input. The first will contain an integer, n ($1 \leq n \leq 10^5$), representing the number of piles of stones Alice has. On the following line will be n integers separated by spaces, where if the ith ($1 \leq i \leq n$) pile has an integer x ($1 \leq x \leq 10^9$), then this indicates that there are 2^x stones in Alice's ith pile. Following this will be a single integer, m ($1 \leq m \leq 10^5$), representing the number of piles of stones Bob has. On the following line will be m integers separated by spaces, where if the jth ($1 \leq j \leq n$) pile has an integer y ($1 \leq y \leq 10^9$), then this indicates that there are 2^y stones in Bob's jth pile.

The Output:

For each day, output one line.

If Alice has more total stones than Bob, output: "`Alice will have more fun!`" If Bob has more stones than Alice, output: "`Bob will have more fun!`" If they have the same number of stones, output: "`Tie!`"

(Sample Input and Sample Output follow on next page)

Sample Input:

```
2
2
100 200
5
101 102 103 104 105
4
2 2 2 2
2
3 3
```

Sample Output:

```
Alice will have more fun!
Tie!
```

Sugar Honeycomb

Filename: `sugar`

Willy Wunko is up to his usual shenanigans in his candy factory. By melting some sugar and adding some baking soda (along with some secret ingredients) he created his latest product: Sugar Honeycombs! To make things more interesting he decided to stamp a shape on each honeycomb and if that shape is successfully carved out, a sweet burst of flavor is magically added. However, if the shape fails to be carved out, a repulsive flavor is added.

Recently you and several other people have won an all inclusive tour of Willy Wunko's factory. At one of the stops on the tour, Willy Wunko lets you all choose a Sugar Honeycomb to try. After seeing the tragedy that befell in the previous room of the tour, you want to ensure that you get the easiest shape to carve out. You notice that shapes with more long and thin portions tend to have a higher perimeter to area ratio and are harder to carve out. Based on this ratio, you want to rank the possible candy shapes from easiest to hardest to carve out.

Due to recent circumstances, the Universal Candy Federation has ruled that all sugar honeycomb shapes must be represented in a rectangular grid of cells (characters) such that all points inside the shape are represented by a '#' and all points outside are represented by a '.'. To gain approval from the Universal Candy Federation, the shape must form one piece (i.e. the shape must be fully connected). The Universal Candy Federation has defined the area of the shape as the number of '#' characters contained within the shape, and defined the perimeter as the minimum number of horizontal or vertical cell edges needed to perfectly encompass that area.

The Problem:

Given a list of shapes represented by a rectangular grid, sort the shapes by their difficulty ratio (perimeter/area).

The Input:

The first line will contain a single, positive integer, t, representing the number of tours you have won. Each tour will start with a line consisting of a single integer, n ($1 \leq n \leq 100$), representing the number of possible shapes for the sugar honeycombs. Each shape will start off with a line containing a string composed strictly of at most 50 lowercase letters representing the name of the shape and integers, r and c ($1 \leq r \leq 100$; $1 \leq c \leq 100$), where r is the number of rows in the grid and c is the number of columns. Each shape will have a unique name. The following r lines will consist of c '`.`' or '`#`' characters. The outermost edges of the grid will be composed of only '`.`' and it is guaranteed that there will be at least one '`#`'.

The Output:

Output n lines where the i^{th} line is the i^{th} easiest candy to carve out. Each line starts with the name of the candy followed by the difficulty ratio formatted as a fraction without any reduction. The fraction format is the perimeter followed by a forward slash followed by the area. If two shapes have the same difficulty ratio, the shape with the lower lexicographical name should be labeled as easier. Output a blank line after the output for each tour.

Sample Input:

```
1
4
triangle 6 9
.........
....#....
...###...
..#####..
.#######.
.........
star 9 9
.........
....#....
....#....
.#######.
..#####..
...###...
..##.##..
..#...#..
.........
circle 8 8
........
...##...
..####..
.######.
.######.
..####..
...##...
........
umbrella 9 9
.........
....#....
..#####..
.#######.
....#....
....#....
....#.#..
....###..
.........
```

Sample Output:

```
circle 24/24
triangle 22/16
star 36/23
umbrella 34/20
```

Zigzag Palindrome

Filename: `zigzag`

Daniel, the acclaimed StringPotato48, was recommended to play this brand new puzzle game themed all around text and word analysis. The first challenge was quite rudimentary: remove all occurrences of the letter 'E' from a word. The next challenge he also found quite easy: find if a word contains another word. Daniel was getting quite bored of these easy challenges so he decided to change to the hardest difficulty level.

He was enjoying it much more until he got to the level one boss: The Reversinator. The Reversinator is an evil robot that could only be shutdown from the Handy Secret Puzzle Terminal. That terminal has a set of puzzles that Daniel must solve in order to shutdown the Reversinator. After messing around with the first puzzle, Daniel realized what the puzzles were asking for: the longest zigzag palindromic substring.

The longest zigzag palindromic substring is the longest consecutive series of characters in a string that obeys two properties. First it must zigzag, meaning that it alternates between strictly increasing and strictly decreasing letters. For instance, "acbfa" is a zigzag string. Since 'c' comes after 'a' alphabetically, it is said to be increasing. Since 'b' comes before 'c' alphabetically, it is said to be decreasing. However, "abca" is not a zigzag string since there are two consecutive increases: 'a' to 'b' and 'b' to 'c'. Also note that a zigzag string can start off increasing or decreasing. Second, it must be palindromic, meaning that it can be read the same forward and backwards. For instance, "racecar" and "noon" are palindromic, but "stat" is not palindromic. So in total the longest zigzag palindromic substring is the longest consecutive set of characters that alternates between strictly increasing and strictly decreasing and is read the same forwards and backwards.

Daniel had no issue solving the first puzzle. He was given "obababo" and quickly identified the solution as "babab". However, due to the change in difficulty, Daniel is now receiving text that is way too long for him to analyze. He has asked you to help him identify the longest zigzag palindromic substring from the text he was given in the puzzle while he deals with the calamity the Reversinator is causing.

The Problem:

Given a set of puzzles, determine the longest zigzag palindromic substring for each puzzle.

The Input:

The first line will contain a single, positive integer, p, representing the number of puzzles to analyze. The next p lines will contain a single string containing less than 1,000 characters. Each string will be composed strictly of lowercase letters and contain no spaces.

The Output:

Output a single line for each puzzle: the longest zigzag palindromic substring for that puzzle. If there are multiple solutions, print the one that occurs earliest in the string.

Sample Input:

```
4
obababo
adsavoluqsqulovaafgdf
dkljfalkfzsusakfafkldhkjydjfkl
djkfaldjfaldjfisgtgtgtgtgsikisgtgtgtgtgsia
```

Sample Output:

```
babab
oluqsqulo
sus
isgtgtgtgtgsikisgtgtgtgtgsi
```