Tenth Annual
University of Central Florida

High School Programming Tournament: Online Edition

Problems – Division 1

Problem Name	Filename
A Man, A Plan, A Canal, Panama	canalanac
Atharva and Chicken	chicken
The Continuing Corgi Conundrum	corgi
Dictionary Ordering	dictionary
Hanukkah Graphs	hanukkah
David's Room Number	room
Super Scrabble	scrabble
Slices	slices
Slime Boss	slime

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

For example, if you are solving Slime Boss,
Call your program file:
slime.c, slime.cpp, slime.java, or slime.py
Call your Java class: slime

A Man, A Plan, A Canal, Panama

Filename: `canalanac`

As you probably know, a palindrome is a string whose alphanumeric characters are the same both forwards and backwards. Perhaps the most famous palindrome of all time is: "A man, a plan, a canal, Panama" that describes Teddy Roosevelt and the building of the Panama Canal. Like every canal designer and palindrome connoisseur, you want to one-up Teddy by building a canal that connects the two halves of Ekalaka Lake (located in Ekalaka, Montana)!

Your canal will consist of at most k level sections of water that span n meters of land. The water level of a section must be an integer strictly greater than the land height anywhere within that section. A section of the canal is a continuous region of water that is entirely the same height. To make building the canal easier, you need to minimize the total height of the water levels across all x coordinates (not necessarily across all sections). Formally, you need to minimize $\Sigma\, w(i)$ where $1 \leq i \leq n$, and $w(i)$ is the water height at x-coordinate i.

Now, since you really want a spot in the Palindromic Hall of Fame (nevermind Mt. Rushmore), you also want the heights of the water in your canal to form a palindrome. That is, $w(i)$ should equal $w(n\text{-}i\text{+}1)$ for all $1 \leq i \leq n$.

Oh, and one last thing: The sum of all water heights across all x values without leading zeros needs to be a palindrome too. Good luck!

The Problem:

Find the minimum sum of the water heights such that your canal has no more than k sections, a side-view of the canal would be palindromic, and the sum of water heights is a palindrome.

The Input:

The first line of input contains a single, positive integer, c, representing the number of canals you want to build. The descriptions of each canal follow. The first line of each description contains two integers n and k ($1 \leq k \leq n \leq 50$), representing the length of land your canal must cross, and the maximum number of sections you are allowed to have, respectively. The next line contains n integers: the heights of the land that boats must be able to go over. All heights are between 1 and 350, inclusive.

The Output:

For each canal, output a single line containing "`Canal #`*`i`*`: `*`t`*" where *`i`* is the canal number in the input (starting with 1) and *`t`* is minimum sum of the water heights required in that canal, such that the water level is strictly higher than the land at every x-coordinate, the canal contains no more than k sections, the canal is palindromic when viewed from the side, and *`t`* is a palindromic integer. If it is impossible to create any palindromic canal that meets the desired conditions, output "`Canal #`*`i`*`: Impossible`" instead.

Sample Input:

```
3
5 3
4 4 1 1 2
5 1
4 4 1 1 2
10 1
2 2 2 1 1 1 2 1 2 1
```

Sample Output:

```
Canal #1: 22
Canal #2: 55
Canal #3: Impossible
```

Atharva and Chicken

Filename: `chicken`

There is one thing about Atharva that everyone knows: Atharva loves chicken! Any mention of chicken gets Atharva incredibly excited. One of his favorite chicken-containing things is the Chick-fil-A chicken sandwich.

However, Atharva is cursed: even though he specifically asks for "no pickles" every time he orders, his sandwich comes with pickles with probability p! This bothers Atharva so much that, despite his love of chicken, after this happens he stays away from Chick-fil-A for d days (including the day he gets pickles).

For example, if $d = 2$ and Atharva's sandwich has pickles on day 6, Atharva will first come back on day 8. Additionally, if there are pickles on Atharva's sandwich, the number of pickles on the sandwich is equiprobably 1, 2, or 3. Atharva goes to Chick-fil-A at most once per day – and unless he has recently gotten pickles he is guaranteed to go to Chick-fil-A every day.

Due to uncontrollable circumstances, every semester d and p change, and sometimes the semester is longer or shorter. This impacts the expected number of pickles Atharva will get in that semester.

The Problem:

Given how many days in the semester, how many days Atharva will stay away from Chick-fil-A when he gets pickles, and the probability of getting pickles on his sandwich each time, determine the expected total number of pickles Atharva will see on his sandwiches all semester.

The Input:

The first line of input contains a single, positive integer, t, representing the number of semesters to process. Each semester will be contained on a new line and will consist of two, positive integers, n ($n \leq 100{,}000$) and d ($d \leq n$), representing the number of days in the semester, and the number of days Atharva will stay away if he gets pickles, respectively, followed by a real number, p ($0 \leq p \leq 1$), representing the probability of getting some amount of pickles on his sandwich.

The Output:

For each semester, output a single real number, e, representing the expected total number of pickles Atharva will find on his sandwiches in the semester. Your answer will be accepted if it is within either 10^{-6} or 0.0001% of the right answer.

Sample Input:

```
2
3 1 1
1000 3 0.1337
```

Sample Output:

```
6
211.04988585090663
```

The Continuing Corgi Conundrum

Filename: `corgi`

Charles, the Corgi-obsessed programmer, is back! He once again visits a pond with exactly n dogs in a nearby field, all of which are Corgis. He wishes to take home exactly k distinct Corgis, and as he is not partial to any particular doggo, he selects k initial dogs at random. However, we all know that Corgis are amazingly energetic and friendly creatures who share inseparable bonds with others of their kind. Thus, upon choosing an initial Corgi, Charles also has to take its friends, and then its friends' friends, and so on...

Now given the friendships that all of the Corgis share, Charles wishes to know what the expected number of Corgis that he will take home is if he chooses k distinct initial Corgis at random.

The Problem:

Given n Corgis, m friendship bonds between them, and a positive integer, k, determine the number of Corgis that Charles can expect to take home, if he picks k distinct initial Corgis at random.

The Input:

The first line contains a single, positive integer, s, which is the number of distinct scenarios to consider. For each scenario, the first line contains three integers, n, m and k ($1 \leq n \leq 1{,}000$; $0 \leq m \leq 1{,}000$; $1 \leq k \leq n$), representing the number of Corgis that exist in the pond, the number of friendship bonds, and the number of Corgis Charles chooses initially, respectively. The following m lines of the scenario will contain friendship bonds, if any, and consist of two integers, u and v ($1 \leq u \leq n$; $1 \leq v \leq n$), meaning that Corgi number u and Corgi number v are friends.

The Output:

For each scenario, output "`Pond #`*`i`*`:` *`x`*" where i is the number of the pond (in the order of the input, starting with 1) and x is the expected number of Corgis that Charles can expect to take with him for this pond to 3 decimal places and rounded. For example, 12.1713 rounds to 12.171, 12.1715 rounds to 12.172, and 12.1718 rounds to 12.172.

(Sample Input and Sample Output follow on next page)

Sample Input:

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

Sample Output:

```
Pond #1: 3.200
Pond #2: 6.543
```

Dictionary Ordering

Filename: `dictionary`

Back when the first dictionary was being written, Dr. Dinkledorf suggested that words should be ordered by the *count* of their letters in alphabetical order. For example, if a word *i* has more occurences of the letter 'a' than the word *j*, then *i* should come before *j*. If they had the same quantity, then if it has more occurences of the letter 'b', 'c', 'd'... and so on.

"But what if the words are anagrams[1]?" asked his assistant, Mr. Binkley. "No worries," Dinkledorf responded, "we will just sort them in lexicographical order[2] in this case."

Of course, this was the worst idea ever, which is why you've never heard of it. Since you would like to demonstrate how bad this idea is to your friends, you plan to write a program that will sort words in Dr. Dinkledorf's ordering scheme.

The Problem:

Sort a list of unique words using Dr. Dinkledorf's ordering scheme.

The Input:

The first line contains a single, positive integer, *t*, representing the number of dictionaries. For each dictionary, multiple lines follow. The first line contains a single integer, *n* ($1 \leq n \leq 10^5$), representing the number of words. Then, *n* lines follow, each consisting of a string of only lowercase letters of positive length up to 20. It is guaranteed that each word is unique.

The Output:

For each dictionary, output *n*+1 lines. The first line to be printed is "`Dictionary #i:`" where *i* is the number of the dictionary in the order of the input (starting with 1). The next *n* lines should each contain a word given in the input, presented in sorted order. Output a blank line after each dictionary.

(Sample Input and Sample Output follow on next page)

[1] Anagram – a word formed by rearranging the letters of another, such as *earth* formed from *heart*.
[2] Lexicographical order – also known as alphabetical order. This is the way words are ordered in modern dictionaries.

Sample Input:

```
3
3
david
lionel
natasha
4
cabbbbb
baaaaaaaa
sarahsara
bbbbbccccc
1
heyguys
```

Sample Output:

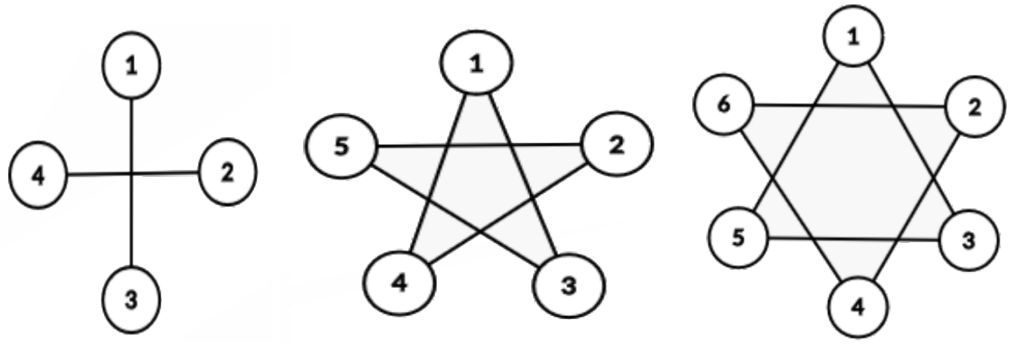
```
Dictionary #1:
natasha
david
lionel

Dictionary #2:
baaaaaaaa
sarahsara
cabbbbb
bbbbbccccc

Dictionary #3:
heyguys
```

Hanukkah Graphs

Filename: `hanukkah`

The judges hope you enjoy your upcoming winter break and have a wonderful holiday season! As you probably know, Christmas *trees* are very popular during the holiday season, but we thought we would instead ask you about Hanukkah *graphs*!

We are defining a Hanukkah graph of size n to be n nodes arranged in a circle, labeled 1 to n in clockwise order, with each node having a bidirectional edge to the node two clockwise and two counterclockwise from itself. The Hanukkah graph of size 6 resembles the Star of David, and is shown below. We would like you to calculate the shortest distance between two given nodes in a Hanukkah graph of size n, or determine that it is impossible to reach the second node from the first traveling only along existing edges.



Hanukkah graphs of sizes 4, 5, and 6

The Problem:

For a Hanukkah graph of size n, calculate the shortest distance from node a to node b, or determine that there is no way to reach one node from the other.

The Input:

The first line of input contains a single, positive integer, g, representing the number of Hanukkah graphs to consider. This is followed by g lines, each containing three integers: n ($3 \le n \le 10^{18}$), a ($1 \le a \le n$), and b ($1 \le b \le n$), representing the number of nodes in the Hanukkah graph, and the two nodes to calculate the distance between, respectively.

The Output:

Output g lines each of the form "`Graph #`x`:` d" where x is the graph number in the input (starting with 1), and d is the shortest distance between the two given nodes or "`Impossible`" if it is impossible to reach one node from the other.

Sample Input:

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

Sample Output:

```
Graph #1: 1
Graph #2: Impossible
Graph #3: 2
Graph #4: 2
```

David's Room Number

Filename: `room`

David, along with a couple of colleagues, has recently travelled outside the country to participate in a programming contest! When he arrived at the hotel, he asked the hotel receptionist for a specific room number. The problem is the receptionist didn't care and gave him a random room. David was outraged, and immediately went online to write a bad Yelp review!

You, a trusted friend of David, noticed his outrage, and asked him what was wrong. After David explained the situation, you looked back at the numbers, and realized that by changing the base of the room number, he could change what value it represented in base 10. So, in an attempt to calm David down, you decided to figure out what base you needed to represent the room number such that it equaled the room number he desired in base 10.

The Problem:

Find a number base *b* such that David's room number in base *b* represents the desired room number.

The Input:

The first line contains a single, positive integer, *t*, representing the number of hotel visits David makes. Each of the next *t* lines will contain two integers, *g* ($10 \leq g \leq 10^6$) and *d* ($10 \leq d \leq 10^{18}$), representing the room number David was given and the room number he desired, respectively.

The Output:

For each hotel visit, output "`Hotel Visit #i: Base b`" each on a separate line, where *i* represents the hotel visit in the input (starting with 1), and *b* represents the base for which David's given room number is equal to his desired room number. A solution is always guaranteed to exist, and *b* is guaranteed to be at least 10.

Sample Input:

```
2
15 18
45 89
```

Sample Output:

```
Hotel Visit #1: Base 13
Hotel Visit #2: Base 21
```

Super Scrabble

Filename: `scrabble`

Alice and Bob are playing a variant of Scrabble called Super Scrabble. In this game, there is a pool of some nonempty subset of the lowercase alphabet that is considered "usable" and a list of n valid words. Alice and Bob take turns picking one of the usable letters and putting a tile with that letter next in the sequence. If at any point, the sequence of placed letters is equal to a valid word, then the last player who played a letter wins and the game is over. If the sequence is longer than every valid word, then the game ends in a draw. There are enough tiles of each letter that they don't need to worry about running out of tiles for any of the usable letters.

Alice will go first. Obviously, both Alice and Bob want to win, and if they cannot win, would rather draw than lose. As usual, Alice and Bob are perfect logicians and will play optimally.

The Problem:

Given the list of usable letters and the valid words, determine the result of the game, assuming Alice plays first and both players play optimally.

The Input:

The first line of input contains a single, positive integer, g, representing the number of games Alice and Bob will play. The description of each game follows. The first line of each description contains two integers, n and m ($1 \leq n \leq 26$; $1 \leq m \leq 10^5$), representing the number of usable letters and the number of valid words, respectively. The next line contains n unique lowercase letters in increasing alphabetic order representing the usable letters. The next m lines each contain a single string of lowercase letters representing a valid word. Each word is at most 10^5 characters long composed out of usable letters, and the sum of all word lengths in a game is at most $2*10^5$.

The Output:

Output g lines each of the form "`Game #`x`: `r" where x is the game number and r is either the message "`Alice wins`", "`Bob wins`", or "`Draw`" depending on the result of the game.

Sample Input:

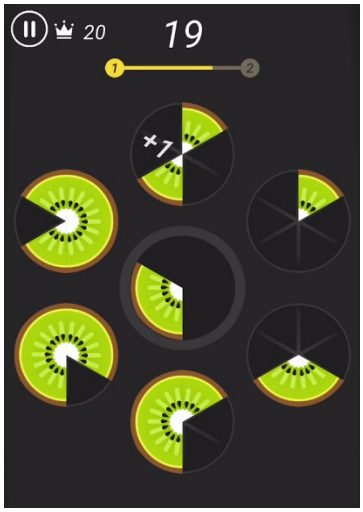
```
3
2 3
ab
aaa
ab
ba
3 4
abc
aaa
ab
cccc
ba
3 3
def
de
dedede
d
```

Sample Output:

```
Game #1: Bob wins
Game #2: Draw
Game #3: Alice wins
```

Slices

Filename: `slices`



Kelly and Jim are obsessed with a new mobile game called Slices! In Slices, the goal of the game is to fill up different circles with slices of fruit and pies while scoring points by placing pieces and completing full circles. After each level, the score required to move on increases.

Kelly and Jim may finish the levels at different times. Sometimes one player gets much further ahead than the other. When this happens, the other player becomes very angry and can't focus on anything in life other than catching up. (In extreme cases of differences between the two friends' progress, one player may forever hate the other.)

Specifically, if there ever is a time that Kelly is more than k levels ahead of Jim, Jim will forever hate Kelly. Similarly, if Jim is ever greater than k levels ahead of Kelly, Kelly will forever hate Jim.

The Problem:

Given the order which Kelly and Jim complete levels in Slices and the maximum number of levels one friend can be ahead of the other without anyone getting angry, determine whether Kelly hates Jim, Jim hates Kelly, they both hate each other, or if their friendship can last another day.

The Input:

The first line contains a single, positive integer, s, representing the number of scenarios to analyze. The first line of each scenario contains two positive integers, n and k ($1 \leq n \leq 1{,}000$; $1 \leq k \leq 1{,}000$), representing the number of events to follow, and the max number of levels one friend can be ahead of the other without putting a strain on their friendship, respectively. The next n lines each contain a single string (either "Kelly" or "Jim") representing who solved a level next and an integer, x ($1 \leq x \leq 1{,}000$), representing which level they just completed. It is guaranteed that the players solve the levels in order starting at 1.

The Output:

For each scenario, if neither friend hates the other, output a single line containing "`Everything is good`". If both players hate each other, output a single line containing "`Their friendship is doomed`". Otherwise, output either "`Kelly hates Jim`" or "`Jim hates Kelly`" depending on which friend hates the other.

Sample Input:

```
3
11 3
Kelly 1
Kelly 2
Jim 1
Kelly 3
Jim 2
Jim 3
Jim 4
Jim 5
Jim 6
Jim 7
Kelly 4
3 2
Jim 1
Kelly 1
Kelly 2
6 1
Kelly 1
Kelly 2
Jim 1
Jim 2
Jim 3
Jim 4
```

Sample Output:

```
Kelly hates Jim
Everything is good
Their friendship is doomed
```

Slime Boss

Filename: `slime`

Sharon the Slayer returns with an all new challenge! This time, he must slay the Slime Boss. The Slime Boss has a special power: every time it is killed, it splits into two slimes, each having half of its health (rounded down). These slimes will continue to split when they are killed in the same way, until the slime has zero health, at which point it will stop splitting. Sharon must slay all slimes. Therefore, while there are slimes alive, Sharon will pick one and attack it. The attacked slime will split (if it can), and all slimes still alive will attack Sharon and deal damage equal to the sum of their health. Determine the minimum damage Sharon will take while fighting the boss, if he kills the slimes in an optimal order.

The Problem:

Find the minimum damage Sharon will take.

The Input:

The input begins with a single, positive integer, t, representing the number of fights. Then, t lines follow, each with a single integer, h ($1 \leq h \leq 50{,}000$), representing the Slime Boss' health.

The Output:

For each fight, print a single line containing "`Fight #`*i*`:` *x*" where *i* is the fight number in the input (starting with 1), and *x* is the minimum amount of damage Sharon will take.

Sample Input:

```
2
4
7
```

Sample Output:

```
Fight #1: 16
Fight #2: 21
```