Thirteenth Annual
University of Central Florida

High School Programming Tournament: Online Edition

Problems – Division 1

Problem Name	Filename
Candy Tax	candy
Life is But a Dream	dream
Hold the Line	line
Mutual Friends	mutual
Porcupine Battles	porcupine
Skipping Stones	stones
Sugar Honeycomb	sugar
King's Table	table
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

Candy Tax

Filename: `candy`

You just finished Trick-or-Treating this Halloween. However, your parents are worried about cavities, so they only let you keep two pieces of candy. Fortunately, you've found a loophole: all the neighbors gave candy out in small goodie bags, to be more sanitary. You reason with your parents that since they're in goodie bags, then all the candy inside the bag counts as one. Your parents begrudgingly agree, but on one condition: you must be able to take the candy from the bags you choose and split them evenly amongst yourself and your siblings.

The Problem:

Given the number of candies in each goodie bag (which could be 0 if you are Tricked!), and the number of people (yourself and your siblings), determine the maximum amount of candy you can keep after selecting 2 bags and evenly dividing the contents amongst yourself and your siblings.

The Input:

The first line of the input will contain a single, positive integer, y, representing the number of years you went Trick-or-Treating. Each year will be described in two lines. The first line of each year will contain two integers, n and s ($0 < n < 10^5$; $0 < s < 10^5$), representing the number of goodie bags and the number of siblings (including yourself), respectively. The next line will contain n integers, a_i ($0 \leq a_i \leq 10^4$), representing the number of pieces of candy in each of your goodie bags.

The Output:

For each year, output a single integer, representing the maximum amount of candy you will be left with after selecting two goodie bags and dividing the candy amongst yourself and your siblings. If there is no way to do this, output 0 (since you won't get any candy).

Sample Input:

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

Sample Output:

```
2
3
0
```

Life is But a Dream

Filename: `dream`

In her dream tonight, Meg is floating down the river on her trusty raft. All of a sudden, she spots a waterfall ahead! The only reasonable thing to do is quickly fashion a jetpack and fly away. As everyone knows, jetpacks are fueled by fireballs. And just like that, n fireballs appear in the air around Meg, each with an intensity value, a_i.

Meg needs to select a subset of exactly k fireballs from the n she is given. To make sure the jetpack flies straight, the chosen subset must obey the property that the sum of its values is equal to the bitwise OR of its values. Meg needs your help before she is swept away: is it possible to construct such a subset, and if so, what is the highest sum that can be achieved?

Note that a subset of an array a is defined as any combination of elements present in a. The bitwise OR operator compares the values (in binary format) of each operand and yields a value whose bit pattern shows which bits in either of the operands has the value 1. For example, if given the values 33 and 13, the bitwise OR would look as follows:

```
   100001 (decimal 33)
OR 001101 (decimal 13)
 = 101101 (decimal 45)
```

The Problem:

Given the intensity values of n fireballs, determine the maximal sum of a subset of size k such that the sum of its values is equal to the bitwise OR of its values.

The Input:

The first line of input contains a single, positive integer, t, representing the number of dreams. For each dream, the first line contains two integers, n and k ($1 \leq k \leq n \leq 2{,}000$), representing the total number of fireballs and the number of fireballs that must be chosen, respectively. The following line contains n integers, a_i ($1 \leq a_i \leq 2{,}000$), representing the intensity value of each fireball.

The Output:

For each dream output a single integer: the maximal sum. If no suitable subset of fireballs exists, print "`It do go down`" instead.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
3
5 1
1 2 4 8 16
3 2
10 2 5
3 3
10 2 5
```

Sample Output:

```
16
15
It do go down
```

Hold the Line

Filename: `line`

Bobby isn't known for his punctuality, and he often finds himself in trouble. Tonight, he woke up and realized he is late for his dinner date! Bobby's girlfriend, tired of his perpetual lateness, is threatening to break up with him if he can't get to the fancy restaurant across town within t minutes. Walking, or even running at top speed, will be too slow this time. Bobby must buy a subway pass!

The town where Bobby lives has n subway stations numbered 1 to n. Subway passes have integer costs starting from 2. A pass of cost k allows access to subway station i if one of the following conditions holds:

- i is equal to 1 (i.e. subway station 1 is accessible with a pass of any cost)
- there exists an integer, x ($2 \leq x \leq k$), such that i % x (the remainder when i is divided by x) is equal to 0

For example, a pass of cost 3 allows travel through subway stations in the set union {1} ∪ {2, 4, 6, 8, 10, ...} ∪ {3, 6, 9, 12, 15, ...}.

There exist m bidirectional subway lines that connect $n + 2$ locations: Bobby's apartment, the restaurant, and the n subway stations. Each subway line takes a certain amount of time to travel. Bobby can take a subway line to a subway station s if and only if station s is accessible with the subway pass he has. If a subway line exists from Bobby's current location to the restaurant, he can always take that line regardless of the cost of his subway pass.

In the interest of saving money while saving his relationship, Bobby wants to buy the subway pass with the lowest cost that allows him to travel from his apartment to the restaurant within t minutes. He needs your help – and fast!

The Problem:

Given the existing subway lines connecting Bobby's apartment, the restaurant, and the n subway stations, determine the smallest k such that Bobby can get to his date within t minutes with a subway pass of cost k.

The Input:

The first line of input contains a single, positive integer, t, representing the number of dates. The first line of each date contains three integers, n ($1 \leq n \leq 200$), m ($1 \leq m \leq 10{,}000$) and t ($1 \leq t \leq 10^9$), representing the number of subway stations, the number of subway lines, and the amount of time Bobby's girlfriend has allotted him, respectively. The following m lines each contain three integers, u ($0 \leq u \leq n + 1$), v ($0 \leq v \leq n + 1$) and w ($1 \leq w \leq 10^9$), meaning that there exists a subway line that connects locations u and v and takes w minutes to travel. Note that the location with index 0 is Bobby's apartment, the locations with indices 1 to n are the n subway stations, and the location with index $n + 1$ is the restaurant.

The input guarantees that no two locations will have more than one subway line connecting them, there will never be a subway line connecting a location with itself, and there will never be a subway line directly connecting Bobby's apartment to the restaurant.

The Output:

Output the smallest value of *k* such that Bobby can get to his date within *t* minutes with a subway pass of cost *k*. If there is no such *k*, print "`Love isn't always on time!`" instead.

Sample Input:

```
3
3 6 10
0 1 11
0 2 4
0 3 4
1 4 10
2 4 7
3 4 6
6 2 10
0 6 9
6 7 1
2 3 1
0 1 1000
1 2 1000
2 3 1000
```

Sample Output:

```
3
2
Love isn't always on time!
```

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
```

Porcupine Battles

Filename: `porcupine`

Two completely different people, Jean and Pierre, have been born and raised in a similar way, adopting and training wild porcupines for battle! Throughout the years, Jean has caught *n* porcupines and Pierre has caught *m* porcupines. Both Jean and Pierre have fed their porcupines fruits, vegetables, milk, chocolate, have hired giraffes, monkeys, bluebirds, tropical fish, hatching chicks, and snails on the slow days to train the stats and levels of their porcupines.

On a bright, sunny day, Jean and Pierre stumbled across each other in the forest. Instead of competing for porcupines to catch, they decided to settle it through a challenge. Jean and Pierre will play a game of porcupine battles! The loser must depart from the premises.

In preparation for the battle, Jean lines up his porcupines on the left side, and Pierre lines up his porcupines on the right side.

The rules of the game are simple:

- A porcupine has an attack, health, and level stat.
- The *middle* is defined as in between the two sides.
- Jean's porcupines are lined up on the left therefore, the porcupine of the highest index that is still alive for Jean is considered closest to the *middle*.
- Pierre's porcupines are lined up on the right therefore, the porcupine of the lowest index that is still alive for Jean is considered closest to the *middle*.
- When a porcupine's health drops below or is equal to 0 at any time during the day, it erupts, *immediately* dealing twice its level in damage to every porcupine (including allied porcupines). Note that this means the death of a porcupine can trigger the deaths of other porcupines and so on.
- At the beginning of a day, and following the end of any chain of eruptions, the living porcupines closest to the *middle* from each team simultaneously attack the opponent's porcupine closest to the *middle*, lowering the striked porcupine's hp by the attacking porcupine's attack.
- The game ends if following a chain of eruptions all of at least one player's porcupines are dead.

The Problem:

Given the arrangement of each player's porcupines, determine who wins, or if it is a draw! A player wins if they have at least one standing porcupine, while all of the opponent's have fallen. Otherwise, it is a draw.

The Input:

The first line of the input will begin with a single, positive integer, *d*, representing the number of days Jean and Pierre met. For each day, the first line contains two positive integers, n ($1 \le n \le 10^5$) and m ($1 \le m \le 10^5$), representing the number of porcupines the first player has, and the number of porcupines the second player has, respectively. Following that are $n+m$ lines each containing three positive integers, a ($1 \le a \le 10^5$), h ($1 \le h \le 10^5$) and l ($1 \le l \le 10^5$), representing the attack, health, and level of each porcupine, respectively. The first n of these lines represent the porcupines of the player on the left. The next m of these lines represent the porcupines of the player on the right..

The Output:

For each day, output a single line containing "`Day #`*i*`: `" where *i* is the number of the day in the input (starting with 1). Followed by "`Jean!`" if Jean won the round, "`Pierre Wins!`" if Pierre won the round, or "`Draw!`" if both players lost all their porcupines.

Sample Input:

```
3
1 2
3 9 1
1 1 2
1 6 1
1 2
3 7 1
1 1 2
1 6 1
1 2
3 5 1
1 1 2
1 7 1
```

Sample Output:

```
Day #1: Jean Wins!
Day #2: Draw!
Day #3: Pierre Wins!
```

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 i^{th} ($1 \leq i \leq n$) pile has an integer x ($1 \leq x \leq 10^9$), then this indicates that there are 2^x stones in Alice's i^{th} 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 j^{th} ($1 \leq j \leq n$) pile has an integer y ($1 \leq y \leq 10^9$), then this indicates that there are 2^y stones in Bob's j^{th} 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 considered 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
```

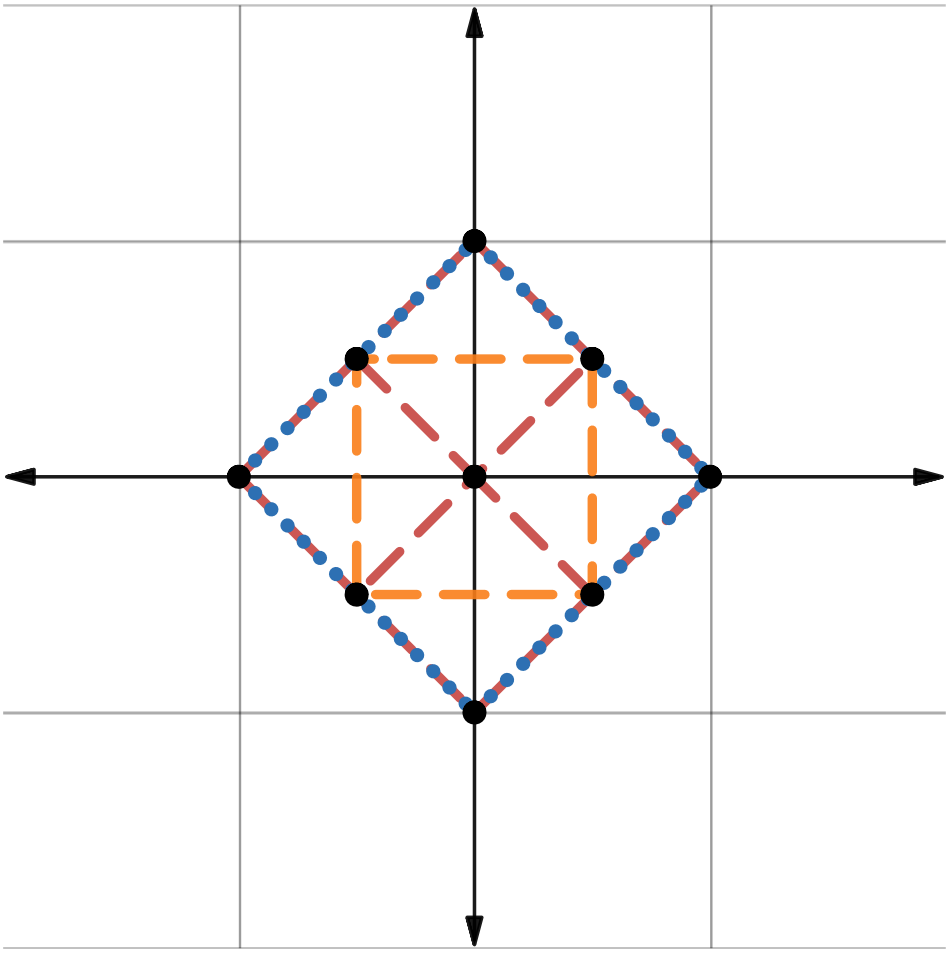
King's Table

Filename: `table`

Circles are overrated! King Arthur, after having his mighty Excalibur defeated by a circular shield, never wants to see a circle again! He has hired you to remodel his meeting room that is filled with numerous circles, but the biggest issue is the circular table sitting in the middle of the room. You have decided to replace this table with the next best thing: A regular polygonal table!

In order for this table to be regular, it must be equilateral and equiangular. Your plan is to build a regular polygonal table such that each side of the table seats exactly one member of the former round table.

After preparing to renovate this table, you remove the outer paneling and the surface of the table, and find a quite bizarre set of table legs. These legs are made of stone (which was not in your original set of materials you had brought) so you can only remove legs, not add legs. You want the new table to be positioned on the remaining legs. You want to now determine how many different ways you can remove legs such that the remaining legs form a regular polygon.



In the above image, there are 9 points (table legs) and we want to find how many regular 4-sided polygons we can make (squares). We have 4 small diagonal squares (red), 1 large diagonal square (blue), and 1 axis-aligned square (orange). This gives a total of 6 squares.

The Problem:

Given a set of leg positions, determine how many regular polygons of a certain number of sides can be made.

The Input:

The first line of the input will be a single, positive integer, t, representing the number of tables to renovate. Each table will start off with a line containing two integers, n and k ($3 \leq n \leq 250$; $3 \leq k \leq 8$), where n is the number of table legs and k is the number of people that must fit around the table. Following will be n lines where the i^{th} line contains two numbers, x_i and y_i ($|x| \leq 1{,}000$; $|y| \leq 1{,}000$), which denotes the x and y coordinate of the i^{th} table leg. x_i and y_i will be a floating point number with at most 4 decimal places. All legs will be distinct points. Two points are distinct if the distance between their x and y coordinates vary by more than 10^{-3}.

The Output:

For each table, output a single integer representing the number of regular polygonal tables of size k that can be made from the n table legs.

Sample Input:

```
2
10 3
-2 0
2 0
0 3.4641
4 3.4641
-4 3.4641
0 -3.4641
-4 0
-4 -3.4641
4 -3.4641
4 0
9 4
0 0
0 2
1 1
-1 1
-2 0
-1 -1
0 -2
2 0
1 -1
```

Sample Output:

```
10
6
```

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
```