Sixteenth Annual
University of Central Florida

High School Programming Tournament: Online Edition

Problems – Division 1

Problem Name	**Filename**
Bounding Bunny	bunny
Cookie Consumption	cookie
Solo Leveling	leveling
Perfect Machine	machine
Picnic Crossing	picnic
Paranormal Plants	plants
Ranged Array	ranged
Sorting Gazebos	sorting
String Splitting	splitting

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

For example, if you are solving Bounding Bunny,
Call your program file:
bunny.cpp, bunny.java, or bunny.py
Call your Java class: bunny

Bounding Bunny

Filename: `bunny`

Benny the Bunny is practicing his leaps! In order to do so, he built an obstacle course consisting of n blocks in a line. Since he is not very consistent, every time Benny jumps forward, he moves forward a uniformly distributed random integer number of blocks between 1 and n - 1 (inclusive).

If there is a chance he jumps beyond the end of the obstacle course, Benny will not jump forward. Instead, he will jump backwards exactly one block until he is sure that he will not land beyond the obstacle course if he jumped forward.

He's asked you to calculate how many leaps he should expect to take before he reaches the last block of the obstacle course from the first block.

The Problem:

Given the number of blocks in the obstacle course, calculate the expected number of jumps Benny will have to take to get from the first block to the last block in the obstacle course.

The Input:

The input consists of a single line containing one integer, n ($2 \leq n \leq 10^4$), representing the number of blocks in the obstacle course.

The Output:

Output one line containing a single real number l: the expected number of leaps Benny will have to take to get to the end of the obstacle course. Your answer will be considered correct if it is within an absolute or relative error of 10^{-4} of the judge's answer.

Sample Input 1:	**Sample Output 1:**
`2`	`1.00000`

Sample Input 2:	**Sample Output 2:**
`8`	`28.00000`

Cookie Consumption

Filename: `cookie`

Cookie Monster (real name Sid Monster) has been given a delicious puzzle by the residents of Sesame Street. There are some plates of cookies in a row. Cookie Monster must choose a consecutive section of plates. As many of the cookies as possible on the selected plates are evenly distributed among the residents, and any remaining cookies on the selected plates are given to Cookie Monster for him to om nom nom. Help Cookie Monster determine the maximum number of cookies he gets to om nom nom.

The Problem:

Given a list of cookie plates, find the maximum number of cookies that Cookie Monster can om nom nom if he gets the remaining cookies on a consecutive section of plates after evenly distributing them among the residents of Sesame Street.

The Input:

The first line of input contains two integers, n and k ($1 \le n \le 10^5$; $1 \le k \le 10^5$), representing the number of plates of cookies and the number of residents of Sesame Street, respectively. The next line of input contains n integers, a_i ($1 \le a_i \le 10^9$), indicating that there are a_i cookies on the ith plate of cookies in the row.

The Output:

Output a single line containing an integer, c: the maximum number of cookies that Cookie Monster can om nom nom.

Sample Input 1:	**Sample Output 1:**

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

Sample Input 2:	**Sample Output 2:**

`4 3`
`3 3 3 3`
	`0`

Solo Leveling

Filename: `leveling`

On his way home from work, Daniel stopped at GameStop and bought a brand new game: Heroic Shadow Prince Three. As a noob, Daniel starts at level 0 and has to attempt quests to change his level. After attempting a quest, his level can change in one of three ways:

- *Complete an Easy Quest*: Daniel doubles his level.
- *Complete a Hard Quest*: Daniel doubles his level, then gains an additional bonus level.
- *Fail a Quest*: Daniel loses half of his levels. Since he cannot have half of a level, his new level is rounded down.

Daniel has decided to 100% the game, which includes unlocking all of the achievements. The i^{th} achievement is unlocked when Daniel reaches level L_i *exactly*. He does not earn the achievement by going over the required level. Note that Daniel can strategically choose to fail quests, and he can complete the achievements in any order. Daniel is also a speedrunner, so he wants to figure out the fewest number of quests he must attempt to unlock all of the achievements. Help Daniel plan his speedrun.

The Problem:

Given the exact level required for each achievement, determine the minimum number of quests Daniel has to attempt to unlock all of the achievements.

The Input:

The first line of input contains a single positive integer, n ($1 \leq n \leq 10^5$), representing the number of achievements. Each of the next n lines contains a single integer, l_i ($1 \leq l_i \leq 10^9$), representing the exact level required to unlock the i^{th} achievement. It is guaranteed that each achievement has a unique level requirement.

The Output:

Output a single integer: the minimum number of quests Daniel has to attempt to unlock all the achievements.

Sample Input 1:

```
2
3
5
```

Sample Output 1:

```
5
```

Sample Input 2:

```
5
7
6
1
4
5
```

Sample Output 2:

```
11
```

Sample Input 3:

```
1
4829546
```

Sample Output 3:

```
23
```

Perfect Machine

Filename: `machine`

Dr. Hunts Mittleshmirtz is up to his schemes again and is plotting to take over the Orlando Metropolitan area. After using the "Eyesore-Inator" to make everything in Orlando an eyesore, he wants to win over the Orlandoan citizens by re-beautifying the city. To do this, he will create the "Perfect-Inator" which is composed of a certain number of critical components.

As his (unpaid) intern, you have been tasked with creating the schematic for his new "Perfect-Inator." He has asked you to model each critical component as a point in the Cartesian plane and to add supports between components which can be modeled as line segments spanning two critical components. However, for the schematic to be physically possible, no two supports can intersect each other at any point other than their endpoints.

Dr. Hunts Mittleshmirtz, however, is concerned about Agent B ruining his plans once again and tampering with his "Perfect-Inator." Instead of removing the self-destruct button, he has asked you to maximize the number of supports in the machine in order to maximize its stability.

The Problem:

Given the number of critical components, design a schematic where you determine where the components go and how they are connected with supports, such that you maximize the number of supports, and no two supports intersect each other outside their endpoints.

The Input:

The input will consist of a single integer, n ($1 \leq n \leq 500$), where n is the number of critical components.

The Output:

First, output n lines, each consisting of two integers, x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$), which denotes the x and y coordinates of the i^{th} component. Then output a line consisting of a single integer, m, denoting the number of supports in the schematic. Finally output m lines consisting of two integers, i and j ($1 \leq i, j \leq n$), denoting that there is a support spanning the i^{th} and j^{th} component. Any valid solution will be accepted.

Sample Input 1:

```
2
```

Sample Output 1:

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

Sample Input 2:

```
3
```

Sample Output 2:

```
0 0
2 0
1 2
3
1 2
2 3
3 1
```

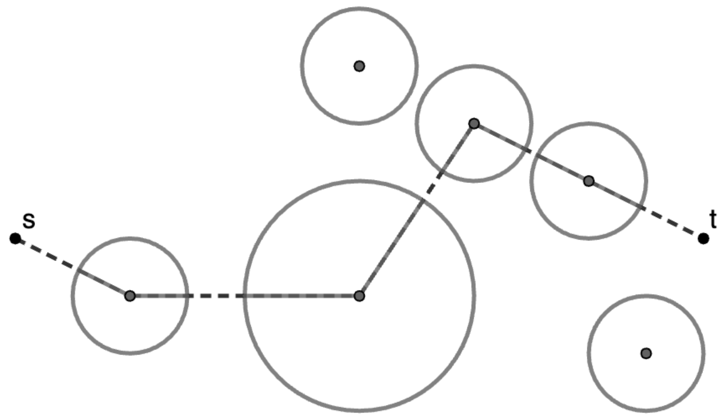
Picnic Crossing

Filename: `picnic`

Frog and Toad are having a picnic. Toad is waiting at their favorite spot across the river, but unfortunately, the bridge they usually take just failed before Frog was able to cross! Frog and Toad won't let this ruin their day, so they've decided to hop their way across the river on a series of lily pads.

Frog will begin his journey at the tip of a nearby pier and attempt to jump from lily pad to lily pad as needed until he makes his way to a point on a pier on the opposite side of the river. Each pad is a perfect circle, and none of the lily pads touch or overlap.

Frog can only hop up to a certain distance between two points (whether they are edges of lily pads or piers). When making a jump, he will hop from a pier or point on the very edge of one lily pad to another pier or the nearest point on the edge of another lily pad; the distance between these points is the distance of the hop. Also, Frog quite likes stability, so he will always cross through the center of a lilypad he is traversing, as pictured below.



Pictured: a possible solution for the sample input. Dashed lines represent hops, while solid lines represent the distance walked by Frog on top of lily pads.

Frog knows the positions and radii of all the lily pads on the river. He has asked you, his Heedful Shortest Path Technician, to determine whether he can make it across, and if so, the minimum total distance he'll need to walk. Frog is a very adept jumper, so he doesn't count hops in the air against this value; only the distance walked atop lily pads.

The Problem:

Given the positions and radii of all lily pads, Frog's starting and ending positions, and his maximum hopping distance, determine the minimum distance he'll need to walk on lily pads to make it across the river (or if it is impossible for him to do so).

The Input:

The first line of input contains two integers, n ($1 \le n \le 1{,}000$) and d ($1 \le d \le 10^6$), representing the number of lily pads and the maximum distance Frog can hop, respectively.

The next line of input contains 4 integers, x_s , y_s and x_t , y_t ($-10^6 \le x_s \le 10^6$; $-10^6 \le y_s \le 10^6$; $-10^6 \le x_t \le 10^6$; $-10^6 \le y_t \le 10^6$), representing the x and y coordinates of the starting and ending positions, respectively. These two points will never lie on or within any lily pad.

Then, n lines follow, each containing three integers, x_i , y_i ($-10^6 \le x_i \le 10^6$; $-10^6 \le y_i \le 10^6$) and r_i ($1 \le r_i \le 10^6$), representing the position and radius of the ith lily pad, respectively. No two lily pads will intersect or overlap.

The Output:

Output a single integer: the minimum distance Frog needs to walk atop lily pads to make it across the river.

If it is impossible for Frog to cross the river, output –1.

Sample Input 1:	**Sample Output 1:**

`6 2`
`0 0 12 0`
`2 -1 1`
`6 -1 2`
`6 3 1`
`8 2 1`
`10 1 1`
`11 -2 1`
	`8`

Sample Input 2:	**Sample Output 2:**

`1 3`
`0 0 12 0`
`6 0 2`
	`-1`

Paranormal Plants

Filename: `plants`

Alien life has just been discovered! A group of scientists has identified a plant species that originates from outside of the solar system, and they now want to grow one of these plants to do further testing on. Unfortunately, the plants seem to interact strangely with Earth's environment. These plants do not grow with water; they can only grow by being treated with a particular chemical compound. However, even this is a complex process.

The scientists have n of these plants arranged in a row, each initially at a height of 1. Each day, only the plants within a given range $[l, r]$ can grow when the compound is applied. Once it is applied, each plant in that range will grow by 1 height unit. However, if any plant in the range is taller than a certain height limit, that plant will absorb all of the compound and none of the plants will be able to grow on that day, including the one that absorbed the compound. The ranges and the height limit are highly unpredictable and change between days.

Fortunately, the scientists have been able to get accurate data on the compound for the next d days. For optimal testing, the scientists would like to grow a single plant as large as they possibly can. They have tasked you with finding the maximum possible height each plant could reach over the next d days. Note that the answer for each plant is being considered independently from the others and you can choose not to apply the compound on a given day.

The Problem:

Given the ranges the compound acts on and the height limit on each of the d days, determine the maximum height each plant could reach.

The Input:

The first line of input contains two integers, n and d ($1 \leq n \leq 3{,}000$; $0 \leq d \leq 3{,}000$), representing the number of plants and the number of days the scientists have data on, respectively. The next d lines each contain three integers, l_i, r_i, and h_i ($1 \leq l_i \leq r_i \leq n$; $1 \leq h_i \leq 3{,}000$), indicating that on the i^{th} day, the compound will grow all plants on the range $[l_i, r_i]$ as long as none of them are taller than a height of h_i.

The Output:

Output n integers, the i-th of which represents the maximum height that the i-th plant could end up being after d days.

Sample Input 1:

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

Sample Output 1:

```
3
3
3
3
3
```

Sample Input 2:

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

Sample Output 2:

```
2
3
3
2
2
```

Ranged Array

Filename: `ranged`

Thomas loves calling things he doesn't like *deranged*. He has decided he needs a term for the opposite phenomenon, so he's coined the term *ranged* to describe things that he enjoys. Thomas considers an array of integers *ranged* if, for every element in the array, the sum of all other elements is evenly divisible by that element.

Can you help Thomas determine whether a given array is ranged?

The Problem:

Given an array of integers, determine whether, for every integer in the array, the sum of all other elements is divisible by that integer.

The Input:

The input begins with a line containing a single integer, n ($2 \leq n \leq 5{,}000$), representing the size of the array. The second line of input consists of n integers, a_i ($1 \leq a_i \leq 10^5$), each representing the i^{th} element in the array.

The Output:

Output a single line. Output "`ranged`" if the array is ranged. Otherwise, output "`deranged`" if it is not.

Sample Input 1:	**Sample Output 1:**

`3`
`2 1 3`
	`ranged`

Sample Input 2:	**Sample Output 2:**

`3`
`2 1 4`
	`deranged`

Sorting Gazebos

Filename: `sorting`

The gazebo business is booming. What started as a curiosity about the word "gazebo" has turned into a multi-billion dollar corporation dedicated to selling gazebos. Thomas, the founder and CEO of this brilliant business, currently has n gazebo models in a line, each having a distinct ranking from 1 to n describing how fancy a given gazebo is (rank 1 is the fanciest). The only problem is the gazebos are not in the right order! Ideally, the gazebos would be sorted in order of fanciness (i.e. from 1 to n) as this will make for the most aesthetically pleasing gazebo lineup. This must be fixed immediately, but due to certain legal contracts Thomas signed with his company's marketing agency, he must use a particular method for rearranging the order of these gazebos.

Each day, Thomas will hire a contractor to reverse the order of some contiguous subsection of the gazebo lineup. Specifically, in a single operation, Thomas will choose two locations l and r in the lineup ($1 \leq l \leq r \leq n$) and reverse the order of the gazebos in the locations from l to r, inclusive. Since contractors are expensive, he would like to sort the gazebos in *at most* $2n$ days.

The Problem:

Given a permutation representing the fanciness of each gazebo, sort them into ascending order (i.e., most fancy to least fancy) using at most $2n$ range reversal operations.

The Input:

The first line of input will contain a single integer, n ($1 \leq n \leq 10^5$) representing the number of gazebos in the lineup. The following line will contain n unique integers, where the ith integer, f_i ($1 \leq f_i \leq n$), represents the fanciness of the ith gazebo.

The Output:

First, output a single line containing, d ($0 \leq d \leq 2n$), the number of days you will hire contractors. For each day, print a single line containing two integers, l and r ($1 \leq l \leq r \leq n$), representing the range of the gazebo lineup the contractors should reverse. After all operations are performed, the gazebo lineup must be sorted in increasing order. If there are multiple possible solutions, any valid solution according to the above constraints will be accepted. Note that you do not need to minimize the number of days used.

Sample Input 1:	**Sample Output 1:**

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

Sample Input 2:	**Sample Output 2:**

`2`
`2 1`
	`1`
	`1 2`

String Splitting

Filename: `splitting`

The Cat has just finished preparing his cake for Thing One and Thing Two's upcoming birthday! Now he has run into another conundrum: the Cat has a single string to divvy up for the two birthday Things. Thing One and Thing Two are very prone to jealousy, so he would like to make sure the strings that each of them receives are identical.

The Cat will first split the string into exactly two contiguous substrings (such that each letter of the original string is in exactly one of the two substrings). Then, he can choose to reverse one of the two strings. The Cat would like to determine if it is possible for him to make two identical strings from these operations.

The Problem:

Given a string, determine whether it is possible to split into two substrings that are identical, possibly after reversing one of the two strings.

The Input:

The input begins with a line containing a single integer, n ($2 \le n \le 10^5$), representing the length of the string. This is followed by a line consisting of a string of n lowercase English letters, representing the string the Cat would like to split.

The Output:

Output a single line. If it's possible for the Cat to split the string as needed, output the message "`Happy Birthday!`" Otherwise, instead output "`No split can fit!`"

Sample Input 1:	Sample Output 1:

`8`
`hspttpsh`
	`Happy Birthday!`

Sample Input 2:	Sample Output 2:

`4`
`baba`
	`Happy Birthday!`

Sample Input 3:	Sample Output 3:

`15`
`redfishbluefish`
	`No split can fit!`