icpc international collegiate programming contest

ICPC North America Contests

Mid-Central USA Regional Contest

Official Problem Set

nac

2022
2023

icpc.foundation

JET BRAINS

icpc global sponsor
programming tools

upsilon pi epsilon
honor society

Mid-Central Regional Programming Contest

25 February 2023

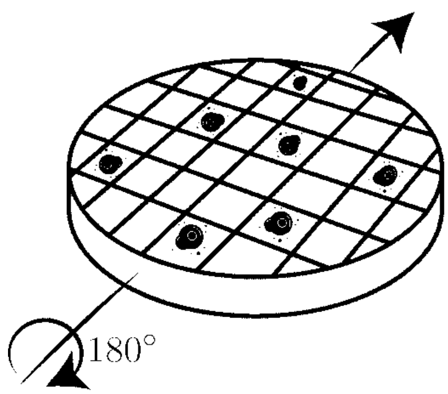
- The languages supported are C, C++ 17 (with Gnu extensions), Java, Python 3 (with pypy3), and Kotlin.
- Python 2 and C# are not supported this year.
- For all problems, read the input data from standard input and write the results to standard output.
- In general, when there is more than one integer or word on an input line, they will be separated from each other by exactly one space. No input lines will have leading or trailing spaces, and tabs will never appear in any input.
- Submit only a single source file for each problem.
- Python may not have sufficient performance for many of the problems; use it at your discretion.

Problem A
Blueberry Waffle
Time Limit: 1 sec

You are using a waffle maker machine to make a delicious blueberry waffle. One side of your waffle is covered in blueberries, while the other side is plain. Initially, the cooking pan of the waffle maker lies horizontally. Once started, the cooking pan will rotate at a constant speed for a fixed duration, then stop. The cooking time is set so that when the waffle maker stops, the cooking pan will not be in a vertical position.

If the cooking pan is not horizontal after this time, the waffle maker will return to a horizontal position via the smallest rotation possible. Therefore, the waffle maker will rotate less than 90 degrees, either forward or backward, until the cooking pan is horizontal again.

The pan rotates at a rate of 180 degrees every r seconds, and stops after f seconds. You don't want to take out your waffle with its blueberry side down. Therefore you'd like to figure out whether the blueberry side of the waffle is up or down after the cooking pan returns to a horizontal position.



Input

The single line of input contains two integers r and f ($1 \leq r, f \leq 10^4$). The pan rotates at a rate of 180 degrees every r seconds, and stops after f seconds. It is guaranteed that after f seconds the cooking pan is not at a vertical position.

Output

Output a single line with a single string, which is `up` if the blueberry side of the waffle is up, or `down` otherwise.

Sample Input 1	Sample Output 1
10 20	up

Sample Input 2	Sample Output 2
10 34	down

Sample Input 3	Sample Output 3
10 47	down

Problem B
Triangle Containment

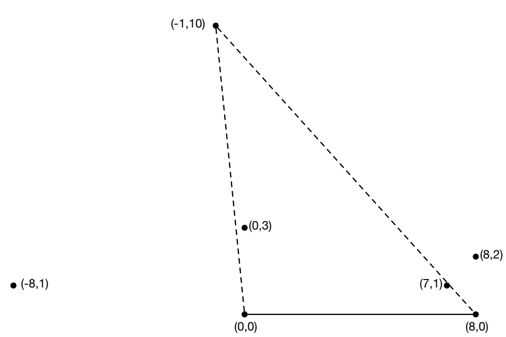
Time Limit: 3 sec

You recently discovered there is treasure buried on your farm land. A **lot** of treasure! You quickly decide to put a fence around the land.

Alas, you have but a single fence post! You will have to drive to town to get more fencing material. But you can't just leave the land as open as it is, so you decide to create a makeshift fence to protect some of the treasure while you are gone. You will place the post in the ground and run some wire in a straight line between two sides of your barn wall and the fence post to section off a triangular area. Also, the ground is very hard: only places that were dug up to bury a treasure are soft enough for you to quickly place the fence post.

To figure out the best option, you first calculate the following. For each of the treasures in your field, if you were to place the fence post at that treasure and complete the fence as described, then what is the total value of all treasures that would be enclosed by the fence? Note that the treasure under the post you place is not considered enclosed by the fence (it might not be safe since someone could dig around the post).

Sample Input 1 is illustrated below. The triangle that includes the point $(-1, 10)$ encloses exactly two other treasure points which have total value $4 + 8 = 12$.



Input

The first line of input contains two integers n ($1 \leq n \leq 10^5$) and x ($1 \leq x \leq 10^9$), where n is the number of treasure points and x fixes the two corners of the barn wall at locations $(0, 0)$ and $(x, 0)$.

Each of the next n lines contains three integers x, y, and v ($-10^9 \leq x \leq 10^9$, $1 \leq y \leq 10^9$, and $1 \leq v \leq 10^9$) giving the location (x, y) and value v of one of the treasure points. All of these points are distinct. It is also guaranteed that for each point, the triangle formed with the barn wall will not contain any other treasure point on its boundary.

Output

Output n lines, one for each treasure point in the order of the input. For each point output a single integer, which is the total value of all points in the interior of the triangle that point forms with the barn wall. Note that the value of the point itself should be excluded from this sum.

Sample Input 1

```
5 8
-8 1 1
-1 10 2
0 3 4
7 1 8
8 2 16
```

Sample Output 1

```
0
12
0
0
8
```

Sample Input 2

```
6 6
0 1 1
2 3 10
2 5 100
3 1 1000
3 5 10000
4 5 100000
```

Sample Output 2

```
0
1000
1010
0
1010
1000
```

Problem C
Everything Is A Nail

Time Limit: 5 sec

As an employee of the Iffy Colossal Pinnacle Construction (ICPC) company building a very tall skyscraper, you have a number of tasks to complete high above the ground in a specific order. You can always choose to skip a task, but you fear that doing so too many times might cause some catastrophic failure of the building. You cannot revisit or complete a task once it has been skipped.

Each task is a nail, a screw, or a bolt. You have three tools: a hammer (works on nails), a screwdriver (works on screws), and a wrench (works on bolts). When you start a new task you can choose to switch your tool out by dropping it (hopefully no one was below you at the time), but when you do so you permanently lose the dropped tool.

Given the list of tasks in the order they should be completed, determine the maximum number of tasks that can be completed. You may choose to use any tool as the initial tool.

Input

The first line of input contains an integer n ($1 \le n \le 3 \times 10^5$), which is the number of tasks you need to complete.

Each of the next n lines contains a single integer t ($0 \le t \le 2$). These are the tasks, in order. Each task is one of 0 (nail), 1 (screw), or 2 (bolt).

Output

Output a single integer, which is the maximum number of tasks that can be completed.

Sample Input 1

```
10
1
1
1
0
0
0
0
2
2
2
```

Sample Output 1

```
10
```

Sample Input 2

```
10
0
1
2
0
1
2
0
1
2
0
```

Sample Output 2

```
5
```

Problem D
Champernowne Count
Time Limit: 1 sec

The nth Champernowne word is obtained by writing down the first n positive integers and concatenating them together. For example, the 10th Champernowne word is “12345678910”.

Given two positive integers n and k, count how many of the first n Champernowne words are divisible by k.

Input

The single line of input contains two integers, n $(1 \leq n \leq 10^5)$ and k $(1 \leq k \leq 10^9)$.

Output

Output a single integer, which is a count of the first n Champernowne words divisible by k.

Sample Input 1	Sample Output 1
`4 2`	`2`

Sample Input 2	Sample Output 2
`100 7`	`14`

Sample Input 3	Sample Output 3
`314 159`	`4`

Sample Input 4	Sample Output 4
`100000 999809848`	`1`

This page is intentionally left blank.

Problem E
Color Tubes

Time Limit: 1 sec

There is a new puzzle generating buzz on social media—Color Tubes. The rules are relatively simple: you are given $n + 1$ tubes filled with $3n$ colored balls. Each tube can hold at most 3 balls, and each color appears on exactly 3 balls (so there are n colors).

Using a series of moves, you are supposed to reach a Color Tubes state—each tube should either hold balls of a single color or it should be empty.

The only move allowed is to take the top ball from one tube and place it into a different tube that has room for it (i.e. holds at most two balls before the move).

You want to write a program to solve this puzzle for you. Initially, you are not interested in an optimal solution, but you want your program to be good enough to solve any puzzle configuration using at most $20n$ moves.

Input

The first line of input contains a single integer n ($1 \leq n \leq 1{,}000$), which is the number of colors.

Each of the next $n + 1$ lines contains three integers b, m and t ($0 \leq b, m, t \leq n$), which are the descriptions of each tube, where b is the color of the ball on the bottom, m is the color of the ball in the middle, and t is the color of the ball on the top.

The tubes are numbered from 1 to $n + 1$ and are listed in order. The colors are numbered from 1 to n. The number 0 describes an empty space. It is guaranteed that no empty space will be below a colored ball.

Output

On the first line output an integer m, the number of moves that your program will use to solve the puzzle. Remember, m has to be at most $20n$.

On the next m lines, output two space-separated integers u and v that describe a move ($1 \leq u, v \leq n + 1$). In each move, you are taking the uppermost ball out of tube u and placing it in tube v, where it will fall until it hits the uppermost ball already in that tube, or the bottom of the tube if the tube is empty.

Your solution will be deemed incorrect if it uses more than $20n$ moves, or any of the moves are not allowed, or the final configuration is not a Color Tubes state.

Sample Input 1

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

Sample Output 1

```
6
3 1
2 3
2 4
3 2
3 2
3 4
```

Sample Input 2

```
1
0 0 0
1 1 1
```

Sample Output 2

```
0
```

Problem F
Food Processor

Time Limit: 1 sec

You have a food processor with a variety of blades that can be attached to it, as well as some food you would like to process into smaller pieces.

The food processor can have one blade attached at any time. Each blade processes food by reducing its average piece size at a particular exponential rate, but it also has a maximum average piece size requirement; if the average piece size of the food is too big for the blade, the food processor will get stuck. Given a starting average food piece size, a target average piece size, and a set of blades for your food processor, determine the minimum amount of processing time needed to process your food into the target average piece size.

Note that we only care about the time spent actively processing food; we do not track time spent switching out blades or loading/unloading the food processor.

Input

The first line of input contains three integers s, t, and n ($1 \le t < s \le 10^6, 1 \le n \le 10^5$), where s is the starting average piece size, t is the target average piece size, and n is the number of blades.

Each of the next n lines contains two integers m and h ($1 \le m, h \le 10^6$). These are the blades, where m is the maximum average piece size of the blade and h is the number of seconds the blade needs to halve the average piece size.

Output

Output a single number, which is the minimum amount of time in seconds needed to process the food to the target average piece size. If it is not possible to reach the target, output -1. Your answer should have a *relative* error of at most 10^{-5}.

Sample Input 1	Sample Output 1
10 1 2 10 10 4 5	23.219281

Sample Input 2

```
10000 9999 1
10000 1
```

Sample Output 2

```
1.4427671804501932E-4
```

Problem G
Digits of Unity
Time Limit: 6 sec

At the beginning of the school year, the students in the International College of Paper Cutters (ICPC) choose their student IDs. The students can choose any positive integer less than or equal to some maximum number for their IDs, but no two students can choose the same student ID.

After some deliberation among the ranks, the students decided they wanted to find some common ground between all their IDs. In particular, they want to choose their IDs such that the bitwise AND of all of their student IDs has at least some minimum number of 1-bits. The students of the ICPC are asking you to write a program to compute the number of ways to do this. Two assignments are different if there is at least one student that has a different student ID in each assignment.

The bitwise AND of two integers a and b is an integer c whose binary representation is as follows: the ith bit of c is 1 if and only if the ith bits of both a and b are 1. C, C++, Java, and Python all support computing the bitwise AND of two integers using the `&` operator.

This definition generalizes to sets of numbers. The bitwise AND of a set of integers S is an integer c whose binary representation is as follows: the ith bit of c is 1 if and only if the ith bit of each element of S is 1.

Input

The single line of input contains three integers n ($1 \le n \le 5 \times 10^5$), k ($1 \le k \le 5 \times 10^5$), and m ($n \le m \le 5 \times 10^6$), where n is the number of students, k is the required minimum number of common bits, and m is the maximum number any student ID could be.

Output

Output a single integer, which is the number of ways to choose n distinct student IDs from the range $[1, m]$ such that the number of 1-bits in the bitwise AND of the student IDs has at least k 1-bits. Since the answer may be large, output it modulo $998{,}244{,}353$.

Sample Explanation

There are 2 students, they want the bitwise AND of their student IDs to have at least 2 1-bits and the maximum allowed student ID is 10. The valid ID assignments are $\{(3, 7), (5, 7), (6, 7), (7, 3), (7, 5), (7, 6)\}$.

Sample Input 1	Sample Output 1
2 2 10	6

Sample Input 2	Sample Output 2
3 4 14	0

Sample Input 3	Sample Output 3
2 1 100000	910073387

Problem H
Hunt the Wumpus
Time Limit: 1 sec

Hunt the Wumpus is a game you play against the computer on a 10×10 grid. At the beginning of the game, four wumpuses are randomly placed on the grid (with no two sharing a location). You need to find and capture all four wumpuses. You guess a square by entering the coordinates as two decimal digits. If you correctly guess the location of a wumpus, you are told you did so, and that wumpus is captured and removed from the grid. Whether or not you hit a wumpus, the Manhattan distance between your guess and the closest wumpus is reported to you. You can use this to locate and find each wumpus. The game ends when the last wumpus is captured.

You have been asked to write the computer portion of the game. User guesses and randomly-generated wumpus locations are defined by a two-digit number, where the high digit is x and the low digit is y for the point (x, y).

To make the game deterministic, we will use our own pseudo-random-number generator, with a supplied seed s. Each random number is generated by first setting $s \leftarrow s + \text{floor}(s/13) + 15$ and then returning the two low digits of s. The first four distinct numbers generated by this process are the locations of the four wumpuses. For a seed of 132, the first location is given by the two low digits of $132 + 10 + 15$, which is 57 and corresponds to the position $(5, 7)$.

Input

The first line contains a single integer s ($10^4 \leq s \leq 10^6$), the seed for the random number generator. Each of the remaining lines contains a two-digit number (possibly with leading zeros). These are the guesses that a player has made, each corresponding to a single grid location.

The input will always be well-formed, and will describe a complete game. The last user guess will always find the last wumpus. There will be at most 250 guesses in any input.

Output

Output the computer's response for each player guess. If a guess hits a wumpus, you should output "`You hit a wumpus!`" on a line. Whether or not the player hit a wumpus, if any wumpuses remain uncaptured, output the Manhattan distance to the closest remaining wumpus. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$, and will therefore always be an integer.

At the end of the game, report the total number of moves m required to locate all the wumpuses by outputting "`Your score is` m `moves.`" on a line.

Sample Input 1

```
203811
00
01
02
03
```

Sample Output 1

```
You hit a wumpus!
1
You hit a wumpus!
1
You hit a wumpus!
1
You hit a wumpus!
Your score is 4 moves.
```

Sample Input 2

```
101628
00
40
60
68
78
95
```

Sample Output 2

```
4
You hit a wumpus!
2
You hit a wumpus!
8
1
You hit a wumpus!
5
You hit a wumpus!
Your score is 6 moves.
```

Problem I
Branch Manager

Time Limit: 4 sec

You are managing a transportation network of one-way roads between cities. People travel through the transportation network one by one in order all starting from the same city, and each person waits for the person before them to stop moving before starting. The people follow a simple algorithm until they reach their destination: they will look at all the outgoing roads from the current city, and choose the one that leads to the city with the smallest label. A person will stop when they either reach their destination, or reach a city with no outgoing roads. If at any point someone fails to reach their destination, the rest of the people still waiting in line will leave.

Before each person enters the transportation network, you can permanently close down any subset of roads to guarantee they reach their destination. The roads that you choose to close down will not be available for future people.

There are n cities, labeled from 1 to n. There are $n - 1$ directed roads, and each road will always be from a lower labeled city to a higher labeled one. The network will form a rooted tree with city 1 as the root. There are m people that want to travel through the network. Each person starts from city 1, and has a specific destination city d in mind. These people will line up in the given order. What is the maximum number of people you can route correctly to their destination if you close roads optimally?

Input

The first line of input contains two integers n and m ($2 \leq n, m \leq 2 \times 10^5$), where n is the number of cities and m is the number of people.

Each of the next $n - 1$ lines contains two integers a and b ($1 \leq a < b \leq n$), denoting a directed road from city a to b. These roads will describe a rooted tree with city 1 as the root.

Each of the next m lines contains a single integer d ($2 \leq d \leq n$), denoting the destination city of the next person in line.

Output

Output a single integer, which is the maximum number of people you can route to the correct destination.

Sample Input 1	Sample Output 1
8 5 1 2 4 8 4 6 1 4 2 5 4 7 2 3 5 2 6 4 8	5

Sample Input 2	Sample Output 2
4 4 1 2 1 3 1 4 3 2 3 4	1

Problem J
Advertising ICPC
Time Limit: 1 sec

You're making a flag to try to advertise ICPC! The flag takes the form of a grid that is already filled with some "C", "I", and "P" letters. A flag is *advertising ICPC* if there exists at least one 2×2 subgrid that looks exactly like the following:

```
IC
PC
```

The flag cannot be rotated or reflected. Every square in the grid must be filled with either a "C", "I", or "P". Count the number of ways to fill the unfilled locations on the flag such that the flag is advertising ICPC.

Input

The first line contains two integers, n and m $(2 \leq n, m \leq 8)$, where n is the number of rows and m is the number of columns in the grid.

The next n lines each contains a string of length m. Each character in the string is either a "C", "I", "P", or "?". A "?" means that that location is not yet filled with a letter.

These n lines form the grid that represents the flag.

Output

Output a single integer, which is the number of ways to fill the flag such that it is advertising ICPC, modulo 998,244,353.

Sample Input 1	Sample Output 1
3 3 ??? ?I? ???	243

Sample Input 2	Sample Output 2

`2 2`
`IC`
`PC`
	`1`

Problem K
Bog of Eternal Stench
Time Limit: 3 sec

You are trying to reach the center of a Labyrinth, which means you must cross the Bog of Eternal Stench. Legend says that if you put so much as one toe in the Bog you will smell bad... forever. You would of course prefer to make it through the Bog with the minimum level of stench possible.

Luckily, you have previously determined that the stench does eventually wear off, and that some areas of the Bog transfer more stench to you than others. There are also small islands where you can rest, without any effect on your stench level. The first island is the starting location of your journey through the Bog of Eternal Stench, and the last is your destination at the other side.

From each island, established bridges allow you to travel to other islands, but these bridges can only be used in one direction. Because this is the Bog of Eternal Stench, traveling along most bridges will increase your overall stench by a specific amount. However, some bridges are quite pleasant, and will decrease your overall stench as you travel along them. But there is a catch—your stench level can never drop below 0. (A bridge that would decrease your stench level below 0 sets it to 0 instead).

You have carefully mapped out all of the islands and bridges, and measured the amount each bridge will increase or decrease your stench. As a result, it may be possible to traverse the Bog of Eternal Stench and emerge with no stench at all!

Your top priority is reaching the destination island with minimum stench; you are willing to take a circuitous path that visits some islands multiple times if doing so achieves this goal. Your path must end at the destination island, but you don't have to leave the Bog immediately the first time you reach your destination, if taking an additional detour and returning to the island later would decrease your final stench value.

Input

The first line of input contains two integers n and m ($1 \leq n, m \leq 2{,}000$), where n is the number of islands and m is the number of direct bridges.

Each of the next m lines contains 3 integers u, v, and s ($1 \leq u, v \leq n$, $-10^9 \leq s \leq 10^9$), indicating that there is a direct bridge from island u to island v that changes your overall stench level by s. It is guaranteed that $u \neq v$, and that there is at most one direct bridge from u to v (but there can also be another direct bridge from v to u).

You may assume that it is possible to reach island n (your destination) from island 1 (your starting location).

Output

Output a single integer, which is the minimum stench level you can exit the Bog with, assuming you begin with 0 stench.

Sample Input 1	Sample Output 1
4 4 1 2 5 1 3 -2 2 4 1 3 4 10	6

Sample Input 2	Sample Output 2
5 5 1 2 1000 2 3 -3 3 4 1 4 2 0 2 5 2	3

Sample Input 3	Sample Output 3
3 3 1 3 -10 3 2 2 2 3 -1	0

Problem L
I Could Have Won
Time Limit: 1 sec

"We will be closing in about 5 minutes. Thank you for visiting the ICPC gym today."

With this announcement, Alice and Bob stopped playing their rock-paper-scissors marathon in the middle of the 10th game. Each player scores a point if their throw beats the other player's throw. Each game was played by the *first-to-11* rule, meaning that whoever scores 11 points first wins the game. Today, Bob narrowly defeated Alice by a single game; he scored 11 points first in five games, while Alice only scored 11 points first in four games.

After carefully inspecting how each game was played, however, Alice realized that she could have won more games than Bob if they played under slightly different rules, such as *first-to-5* or *first-to-8*, instead of the regular *first-to-11*.

Given the sequence of points scored by Alice and Bob, determine all values of k such that Alice would have won more games than Bob under the *first-to-*k rule.

Both Alice and Bob start with zero points at the beginning of a game. As soon as one player reaches k points, that player wins the game, and a new game starts. Alice wins a game if she scores k points before Bob does. Neither player wins the game if it's interrupted by the gym closing before either player reaches k points.

Input

The single line of input consists of a string of uppercase letters "`A`" or "`B`", denoting who scored each point from the beginning of the rock-paper-scissors marathon. The length of the string is between 1 and $2{,}000$ letters, inclusive. "`A`" means Alice scored the point, "`B`" means Bob scored the point.

Output

On the first line, output the number of positive integers k for which a *first-to-*k rule would have made Alice win more games than Bob. If this number isn't zero, on the next line output all such values of k in increasing order, separated by spaces.

Sample Input 1	Sample Output 1
BBAAABABBAAABB	3 3 6 7

Sample Input 2	Sample Output 2
AABBBAAB	2 2 4

Problem M
Creative Accounting
Time Limit: 5 sec

When accounting for the profit of a business, we can divide consecutive days into fixed-sized segments and calculate each segment's profit as the sum of all its daily profits. For example, we could choose seven-day segments to do our accounting in terms of weekly profit. We also have the flexibility of choosing a segment's starting day. For example, for weekly profit we can start a week on a Sunday, Monday, or even Wednesday. Choosing different segment starting days may sometimes change how the profit looks on the books, making it more (or less) attractive to investors.

As an example, we can divide ten consecutive days of profit (or loss, which we denote as negative profit) into three-day segments as such:

$$3, 2, -7 \mid 5, 4, 1 \mid 3, 0, -3 \mid 5$$

This gives us four segments with profit $-2, 10, 0, 5$. For the purpose of this division, partial segments with fewer than the fixed segment size are allowed at the beginning and at the end. We say a segment is profitable if it has a strictly positive profit. In the above example, only two out of the four segments are profitable.

If we try a different starting day, we can obtain:

$$3, 2 \mid -7, 5, 4 \mid 1, 3, 0 \mid -3, 5$$

This gives us four segments with profit $5, 2, 4, 2$. All four segments are profitable, which makes our business look much more consistent.

You're given a list of consecutive days of profit, as well as an integer range. If we can choose any segment size within that range and any starting day for our accounting, what is the minimum and maximum number of profitable segments that we can have?

Input

The first line of input has three space-separated integers n, ℓ and h ($1 \leq \ell \leq h \leq n \leq 3 \times 10^4$, $h - \ell \leq 1{,}000$), where n is the number of days in the books, ℓ is the minimum possible choice of segment size, and h is the maximum possible choice of segment size.

Each of the next n lines contains a single integer p ($-10^4 \leq p \leq 10^4$). These are the daily profits, in order.

Output

Output on a single line two space-separated integers min and max, where min is the minimum number of profitable segments possible, and max is the maximum number of profitable segments possible. Both min and max are taken over all possible choices of segment size between ℓ and h and all possible choices of starting day.

Sample Input 1	Sample Output 1
10 3 5 3 2 -7 5 4 1 3 0 -3 5	2 4