South African Computer Olympiad
Training Camp 3, 2006
Day 1

Overview

Author	**Max Rabkin**	**Joshua Yudaken**	**Keegan Carruthers-Smith**	**Ralf Kistner**
Problem	**inscript**	**robot**	**stone**	**meeting**
Source	inscript.c inscript.cpp inscript.pas	robot.c robot.cpp robot.pas	stone.c stone.cpp stone.pas	meeting.c meeting.cpp meeting.pas
Input file	inscript.in	robot.in	stone.in	meeting.in
Output file	inscript.out	robot.out	stone.out	meeting.out
Time limit	2 seconds	1 second	2 seconds	1 second
Number of tests	10	10	10	10
Points per test	10	10	10	10
Total points	**100**	**100**	**100**	**100**

The maximum total score is 400 points.

South African Computer Olympiad
Training Camp 3, 2006
Day 1

Ancient Inscriptions

Author

Max Rabkin

Introduction

At Oklo, Gabon, there is a large deposit of "enriched" uranium. It is generally thought that this is purely natural and coincidental, but some believe that it may be the remains of an ancient civilisation's power station. Archeologists working nearby have discovered several inscriptions. They believe that these may have been written by the power station's engineers, but need your help to prove it.

The inscriptions are encrypted using an "alternation cipher": the writer would start writing from either the left or right, and then change direction, always filling in towards the centre, changing direction an arbitrary number of times (that number is the "alternation count", and may be zero). Since anyone who knows anything keeps a backup of their data, there are two copies of each inscription (usually encoded differently, but with exactly the same punctuation and capitalisation). It is in decryption that the archeologists need help.

Task

Write a program that, given the two inscriptions, determines whether they encode the same message, and if so finds the sum of the alternation counts which could transform the unencrypted text ("plaintext") into the respective encrypted forms.

Example

Consider the plaintext "URANIUM". It may be encoded as follows (the dots, arrows and numbers are for clarity, and are not part of the inscription).

Starting from the left:	$\overset{1}{\underset{\rightarrow}{}}$ `UR.....`
Then switching directions:	$\overset{1}{\underset{\rightarrow}{}}$ $\overset{2}{\underset{\leftarrow}{}}$ `UR..INA`
Back to left-to-right:	$\overset{1}{\underset{\rightarrow}{}}$ $\overset{3}{\underset{\rightarrow}{}}$ $\overset{2}{\underset{\leftarrow}{}}$ `URUMINA`

So "URUMINA" decodes to "URANIUM" with an alternation count of two (since the writer changed direction twice). If the other inscription was "RANIUMU" (with alternation count 1), then the sum of alternation counts a_{tot} for the plaintext "URANIUM" is 3. There may be other possible plaintexts with the same alternation count sum.

Input (inscript.in)

The input consists of two lines of length N, each containing one encrypted version of the text. Both strings consist of upper- and lowercase letters, underscores, full stops and the digits 0–9.

Sample input

```
URUMINA
RANIUMU
```

Output (inscript.out)

The output consists of one line containing one item. If the two inscriptions could not encode the same plaintext, that item is the string "`INCOMPATIBLE`". Otherwise, the item is a single non-negative integer, $a_{\text{tot}} = a_1 + a_2$, where a_1 and a_2 are the alternation counts from the plaintext to the respective inscriptions. There may be more than one possible plaintext, in which case the program must output the minimum a_{tot}.

Sample output

```
3
```

Constraints

- $1 \leq N \leq 400$

40% constraints

- $1 \leq N \leq 50$

Time limit

2 seconds.

Scoring

Correct answers will score 100%; incorrect or incorrectly formatted answers will score 0%.

South African Computer Olympiad
Training Camp 3, 2006
Day 1

Robot Run

Author

Joshua Yudaken

Introduction

Dave has a natural love for traffic lights... unfortunately only when they are green, and his life mission has become avoiding red traffic lights. Unfortunately, Dave has found himself in a bit of a pickle. He needs to reach the orange store as quickly as possible without the sight of a single red traffic light and while dodging obstacles.

Task

Help Dave move from the top left corner of the traffic grid to the bottom right (where the orange store is located) without standing on a point from which he can see a red traffic light. Dave can see all the traffic lights horizontally or vertically from himself, as long as there are no obstacles between him and the robot. The traffic lights do not have a yellow phase and all change from red to green (or vice-versa) at the same time, every K seconds. "Obstacles" to Dave's vision and movement consist of scattered stone blocks and also the traffic lights themselves. Dave himself can wait (not move), or move up, down, left, or right. He must never be on a "forbidden" square, where a square is forbidden if it is an obstacle, or is an empty square from which he can see a red traffic light.

Example

Take $K = 2$ and the map below, where "#" is a stone block, "R" is a robot starting red and "G" is a robot starting green. Coordinates are (x, y) where $(1, 1)$ is the top left corner (this is a flip of the usual $(row, column)$ coordinates). The following is a diagrammatic representation of the map at $T = 0$:

```
......
......
.R.##.
..G...
```

Applying the robot visibility rules, we can mark all the forbidden squares with "#":

```
.#....
.#....
#####.
.##...
```

Every $K = 2$ time steps, the robots change to the following alternate state of forbidden squares (and change back to the original K steps later):

```
..#...
..#...
.####.
######
```

Optimal solutions take 9 steps for Dave to reach the orange shop. One solution is described below, where "original" and "alternate" refer to the two states of the robots.
$T = 0$ (original): At $(1, 1)$ (starting position). Move down.
$T = 1$ (original): Arrive at $(1, 2)$. Move right.
$T = 2$ (alternate): Arrive at $(2, 2)$ (which is now clear). Wait.
$T = 3$ (alternate): Still at $(2, 2)$. Move right.
$T = 4$ (original): Arrive at $(3, 2)$ (which is now clear). Move right.
$T = 5$ (original): Arrive at $(4, 2)$. Move right.
$T = 6$ (alternate): Arrives at $(5, 2)$. Move right.
$T = 7$ (alternate): Arrive at $(6, 2)$. Move down.
$T = 8$ (original): Arrive at $(6, 3)$. Move down.
$T = 9$ (original): Arrive at $(6, 4)$ (the orange shop). Dave can now buy his oranges.

Input (robot.in)

The first line of the input contains a five space-separated integers: W, H, K, S and R (the width, height, robot timing, number of stone blocks and number of robots respectively). The next S lines contain two space-separated integers, x and y, representing the horizontal and vertical position of each stone block. The next R lines contain two space-separated integers followed by a space and a character c, representing the horizontal and vertical positions of a red robot if c is "R" or a green robot if c is "G".

Sample input

```
6 4 2 2 2
4 3
5 3
2 3 R
3 4 G
```

Output (robot.out)

A single line with the earliest time at which Dave can arrive the orange shop or the word "none" if it is not possible.

Sample output

9

Constraints

- $1 \leq W, H \leq 500$
- $1 \leq K \leq 6$
- $0 \leq S, R \leq 500$

50% constraints

- $0 \leq R \leq 25$

Time limit

1 second.

Scoring

An optimal answer will score 100%, while a sub-optimal or invalid answer will score 0%.

South African Computer Olympiad
Training Camp 3, 2006
Day 1

Stone

Author

Keegan Carruthers-Smith

Introduction

You are on the IT staff for the hit TV show Survivor XXI: Merida. This week's challenge is to represent a fixed length message using black and white stones. The stones are placed next to each other in a line. The winner is the contestant who uses the least amount of stones. The producers of the show want to know what is the least amount of stones a contestant could use.

Task

You are provided with a histogram of the characters in the message. (This is a frequency count of each character). Due to the nature of the challenge, no character's representation in black and white stones can be a prefix of another character's representation. You need to output what the least amount of stones a contestant could use.

Example

The message "an aardvark" would have a histogram of:
(space) — 1
a — 3
n — 1
r — 2
d — 1
v — 1
k — 1

A possible encoding would be (1 is a black stone and 0 is a white stone):
(space) — 1011
a — 11
n — 1010
r — 01
d — 100
v — 000
k — 001

This will make the layout of the stones: 11101010111111010000011001 This gives a length of 27, which is minimal.

Input (stone.in)

The first line contains the number of characters in the histogram, N. The next N lines each contain a single integer, F_i, which represents the frequency of a character.

Sample input

```
7
3
1
1
2
1
1
1
```

Output (stone.out)

The output file contains a single integer, M, the minimum length of the encoded message.

Sample output

```
27
```

Constraints

- $1 \leq N \leq 200000 = 2 \times 10^5$
- $1 \leq F_i \leq 2000000000 = 2 \times 10^9$

60% constraints

- $1 \leq N \leq 2000$
- $1 \leq F_i \leq 10000$

40% constraints

- $1 \leq N \leq 10$
- $1 \leq F_i \leq 600$

Time limit

2 seconds.

Scoring

You will score 100% for a test case if you output the minimum amount of stones required, otherwise you will receive 0.

Meeting

Author

Ralf Kistner

Introduction

A group of people are going to a meeting. Every person in the group, apart from the leader, has exactly one superior, which is also in the group. The leader has no superiors. Whenever two people with the same superior arrive at the meeting (not necessarily directly after each other), their superior has to arrive directly afterwards (after the second one). The superior may arrive earlier, but not later.

Task

Given a list of the people and their superiors, find the maximum number of people that can arrive before the leader.

Example

There are 5 people going to the meeting, numbered 0 to 4. Person 0 is the leader. 1, 2 and 3 have 0 as their superior, and 4 has person 2 as his superior.

A maximum of 2 people from persons 1, 2 and 3 can arrive before the leader. Person 4 can arrive before any one of them. This gives the maximum number of people that can arrive before the leader as 3. For example: first person 1, 4 and 2 arrive. Then the leader has to arrive directly after 4, and then finally 3 arrives. There are many other orderings that they can arrive in, but they will all give a maximum of 3 people that arrive before the leader.

Input (meeting.in)

The first line of the file contains one integer, N, the number of people in the group. The next $N - 1$ lines each contains the superior of a person: the first line will contain the superior of person 1, the second line of person 2, etc. The leader is person 0.

Sample input

```
5
0
0
0
2
```

Output (meeting.out)

The first line of the output contains one integer, the maximum number of people that can arrive before the leader.

Sample output

```
3
```

Constraints

- $1 \leq N \leq 20000$
- There will be no cycles in the input

50% constraints

- $1 \leq N \leq 50$

Time limit

1 second.

Scoring

100% for a correct answer, 0% for an incorrect answer.