SILVER PROBLEMS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Four problems numbered 6 through 9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 6: Hide and Seek [Jacob Steinhardt, 2009]

Bessie is playing hide and seek (a game in which a number of players

hide and a single player (the seeker) attempts to find them after

which various penalties and rewards are assessed; much fun usually

ensues).

She is trying to figure out in which of N (2 <= N <= 20,000) barns

conveniently numbered 1..N she should hide. She knows that FJ (the

seeker) starts out in barn 1. All the barns are connected by M (1

<= M <= 50,000) bidirectional paths with endpoints A\_i and B\_i (1

<= A\_i <= N; 1 <= B\_i <= N; A\_i != B\_i); it is possible to reach

any barn from any other through the paths.

Bessie decides that it will be safest to hide in the barn that has

the greatest distance from barn 1 (the distance between two barns

is the smallest number of paths that one must traverse to get from

one to the other). Help Bessie figure out the best barn in which

to hide.

PROBLEM NAME: hideseek

INPUT FORMAT:

\* Line 1: Two space-separated integers: N and M

\* Lines 2..M+1: Line i+1 contains the endpoints for path i: A\_i and

B\_i

SAMPLE INPUT (file hideseek.in):

6 7

3 6

4 3

3 2

1 3

1 2

2 4

5 2

INPUT DETAILS:

The farm layout is as follows:

1--2--5

| /|

|/ |

3--4

|

6

OUTPUT FORMAT:

\* Line 1: On a single line, print three space-separated integers: the

index of the barn farthest from barn 1 (if there are multiple

such barns, print the smallest such index), the smallest

number of paths needed to reach this barn from barn 1, and the

number of barns with this number of paths.

SAMPLE OUTPUT (file hideseek.out):

4 2 3

OUTPUT DETAILS:

Barns 4, 5, and 6 are all a distance of 2 from barn 1. We choose barn 4

because it has the smallest index.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 7: Cow Line [Jacob Steinhardt, 2009]

Farmer John's N cows (conveniently numbered 1..N) are forming a

line. The line begins with no cows and then, as time progresses,

one by one, the cows join the line on the left or right side. Every

once in a while, some number of cows on the left or right side of

the line all leave the line to go graze in their favorite pasture.

FJ has trouble keeping track of all the cows in the line. Please

help him.

The cows enter the line in numerical order 1..N, and once a cow

leaves the line she never re-enters it. Your program will be given

S (1 <= S <= 100,000) input specifications; each appears on a single

line and is one of two types:

\* A cow enters the line (a parameter indicates whether on the

left or right).

\* K cows leave the line from the left or right side (supplied

parameters define both the number of cows and which side).

Input lines never request an operation that can not be performed.

After all the input lines have been processed, your program should

print the cows in the line in order from left to right. The final

line is guaranteed to be non-empty at the end of the input

specifications.

PROBLEM NAME: cline

INPUT FORMAT:

\* Line 1: A single integer: S

\* Lines 2..S+1: Line i+1 contains specification i in one of four

formats:

\* A L -- a cow arrives on the Left of the line

\* A R -- a cow arrives on the Right of the line

\* D L K -- K cows depart the Left side of the line

\* D R K -- K cows depart the Right side of the line

SAMPLE INPUT (file cline.in):

10

A L

A L

A R

A L

D R 2

A R

A R

D L 1

A L

A R

OUTPUT FORMAT:

\* Lines 1..??: Print the numbers of the cows in the line in order from

left to right, one number per line.

SAMPLE OUTPUT (file cline.out):

7

2

5

6

8

OUTPUT DETAILS:

Input Resulting Cow Line

A L 1

A L 2 1

A R 2 1 3

A L 4 2 1 3

D R 2 4 2

A R 4 2 5

A R 4 2 5 6

D L 1 2 5 6

A L 7 2 5 6

A R 7 2 5 6 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 8: Cow Digit Game [Neal Wu, 2007]

Bessie is playing a number game against Farmer John, and she wants

you to help her achieve victory.

Game i starts with an integer N\_i (1 <= N\_i <= 1,000,000). Bessie

goes first, and then the two players alternate turns. On each turn,

a player can subtract either the largest digit or the smallest

non-zero digit from the current number to obtain a new number. For

example, from 3014 we may subtract either 1 or 4 to obtain either

3013 or 3010, respectively. The game continues until the number

becomes 0, at which point the last player to have taken a turn is

the winner.

Bessie and FJ play G (1 <= G <= 100) games. Determine, for each

game, whether Bessie or FJ will win, assuming that both play perfectly

(that is, on each turn, if the current player has a move that will

guarantee his or her win, he or she will take it).

Consider a sample game where N\_i = 13. Bessie goes first and takes

3, leaving 10. FJ is forced to take 1, leaving 9. Bessie takes the

remainder and wins the game.

PROBLEM NAME: cdgame

INPUT FORMAT:

\* Line 1: A single integer: G

\* Lines 2..G+1: Line i+1 contains the single integer: N\_i

SAMPLE INPUT (file cdgame.in):

2

9

10

OUTPUT FORMAT:

\* Lines 1..G: Line i contains "YES" if Bessie can win game i, and "NO"

otherwise.

SAMPLE OUTPUT (file cdgame.out):

YES

NO

OUTPUT DETAILS:

For the first game, Bessie simply takes the number 9 and wins.

For the second game, Bessie must take 1 (since she cannot take 0), and then

FJ can win by taking 9.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 9: Grazing2 [Osman Ay and Jacob Steinhardt, 2005]

Farmer John has N (2 <= N <= 1,500) prize milk cows conveniently

numbered 1..N. His newly-painted barn has S (N <= S <= 1,000,000)

stalls (conveniently numbered 1..S) in a single long line; each

stall is a unit distance from its neighboring stall(s).

The cows have made their way to the stalls for a rest; cow i is in

stall P\_i. Antisocial as they are, the cows get grumpy if they are

situated in stalls very close to each other, so Farmer John wants

to move the cows to be as spread out as possible.

FJ wants to make sure that the N - 1 distances between adjacent

cows are as large as possible, and he would also like them to be

similar to each other (i.e., close to equi-distant spacing).

In particular, FJ would like all distances between adjacent cows

to be at most 1 different from (S - 1) / (N - 1), where integer

division is used. Moreover, he would like as many of these distances

as possible to be exactly equal to (S - 1) / (N - 1) [integer

division]. Thus, with four cows and eight stalls, one can place the

cows at positions 1, 3, 5, 8 or 1, 3, 6, 8 but not at 1, 2, 4, 7

or 1, 2, 4, 8.

Help FJ spread the cows as efficiently as possible by calculating

and reporting the minimum total distance that the cows have to move

in order to achieve proper spacing. Ignore the distance it takes

for a cow to enter or exit a stall.

PROBLEM NAME: graze2

INPUT FORMAT:

\* Line 1: Two space-separated integers: N and S

\* Lines 2..N+1: Line i+1 contains the single integer: P\_i

SAMPLE INPUT (file graze2.in):

5 10

2

8

1

3

9

INPUT DETAILS:

1 2 3 4 5 6 7 8 9 10

Cow Locs | A | B | C | . | . | . | . | D | E | . |

OUTPUT FORMAT:

\* Line 1: A single integer: the minimum total distance the cows have

to travel. This number is guaranteed to be under 1,000,000,000

(thus fitting easily into a signed 32-bit integer).

SAMPLE OUTPUT (file graze2.out):

4

OUTPUT DETAILS:

Cows move from stall 2 to 3, 3 to 5, and 9 to 10. The total distance

moved is 1 + 2 + 1 = 4. The final positions of the cows are in

stalls 1, 3, 5, 8, and 10.

1 2 3 4 5 6 7 8 9 10

Init Stall | A | B | C | . | . | . | . | D | E | . |

Final Stall | A | . | B | . | C | . | . | D | . | E |

Distance moved | 0 | . | 1 | . | 2 | . | . | 0 | . | 1 |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*