XXXIV Maratón Nacional de Programación Colombia - ACIS / REDIS / CCPL 2020 ICPC

Problems - Warmup

(This set contains 4 problems; problem pages numbered from 1 to 7)

General Information. Unless otherwise stated, the conditions stated below hold for all the problems. However, some problems might have specific requirements and it is important to read the problem statements carefully.

Program name. Each source file (your solution!) must be called

`<codename>.c`, `<codename>.cpp`, `<codename>.java`, or `<codename>.py`

as instructed below each problem title.

Input.

1. The input must be read from the standard input.
2. In most problems, the input can contain several test cases. Each test case is described using a number of lines that depends on the problem.
3. In most cases, when a line of input contains several values, they are separated by single spaces. No other spaces appear in the input. There are no empty lines.
4. Every line, including the last one, has the usual end-of-line mark.
5. If no end condition is given, then the end of input is indicated by the end of the input stream. There is no extra data after the test cases in the input.

Output.

1. The output must be written to the standard output.
2. The result of each test case must appear in the output using a number of lines, which depends on the problem.
3. When a line of results contains several values, they must be separated by single spaces. No other spaces should appear in the output. There should be no empty lines.
4. Every line, including the last one, must have the usual end-of-line mark.
5. After the output of all test cases, no extra data must be written to the output.
6. To output real numbers, if no particular instructions are given, round them to the closest rational with the required number of digits after the decimal point. Ties are resolved rounding to the nearest upper value.

A: **Dominoes Magic Squares**

Source file name: `dominoes.c,` `dominoes.cpp`, `dominoes.java`, *or* `dominoes.py`
Author: `Rodrigo Cardoso`

A *domino set* is a collection of tiles of the form

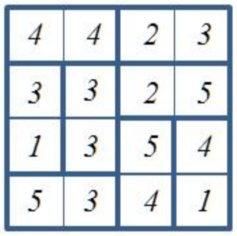
$$[a \mid b]$$

with integer labels a and b satisfying $0 \leq a, b \leq 6$. Both $[a \mid b]$ and $[b \mid a]$ are descriptions of the same domino tile. A complete domino set has exactly 28 tiles and the sum of all its labels is 168.

A *magic square* is a square of integer numbers whose rows, columns, and diagonals have the same sum. Since domino tiles can be seen as planar objects of 2 unit squares, they can be used to build magic squares. For instance, the set of domino tiles

$$[1 \mid 4], [5 \mid 2], [4 \mid 4], [2 \mid 3], [5 \mid 4], [5 \mid 3], [1 \mid 3], [3 \mid 3]$$

can be arranged into a magic square of side 4 units with rows, columns, and diagonals adding up to 13:



However, it is impossible to build a 4×4 magic square with the following set of titles adding up to 15 in rows, columns, and diagonals:

$$[6 \mid 5], [2 \mid 4], [2 \mid 2], [5 \mid 5], [5 \mid 4], [5 \mid 1], [2 \mid 3], [3 \mid 6].$$

Assume you are given 8 domino tiles: can you arrange them into a 4×4 magic square?

Input

The input consists of several test cases. A test case comprises 8 consecutive lines of input, each one containing two blank-separated integers a and b, $0 \leq a, b \leq 6$, representing the tile $[a \mid b]$. You can assume that a test case does not contain repeated dominoes.

The input must be read from standard input.

Output

For each test case, output one line with the unique character '`Y`' if a magic square can be built with the given domino tiles and '`N`' otherwise.

The output must be written to standard output.

Sample Input

```
1 4
5 2
4 4
2 3
5 4
5 3
1 3
3 3
6 5
2 4
2 2
5 4
5 5
5 1
2 3
3 6
```

Sample Output

```
Y
N
```

B: Peanoland contacting Gaussland

Source file name: `gauss.c,` `gauss.cpp`, `gauss.java`, *or* `gauss.py`
Author: `Rafael García`

When scientists in Peanoland discovered that the remote Gaussland planet was inhabited they rejoiced at knowing that they were not alone in the universe. Likewise, there were widespread celebrations in Gaussland when they discovered life in planet Peanoland. Joy did not last long as both planets realized how difficult it was for them to communicate.

After many experiments they discovered that the best way to communicate was through basic light impulses and each planet independently developed its own ACIS table (think of it like an ASCII table but for interplanetary communications). Unfortunately, the ACIS table was not working and scientists simply could not figure out what they had done wrong.

Enter the brilliant Dr. Albert C. Munchhausen whom, through a far bit of trial and error, noticed that communication problems were due to the usage of two different numerical systems. In Peanoland, they were using natural numbers $\mathbb{N}$ in their ACIS table. Gausslanites were instead using Gaussian integers, i.e. the set $\mathbb{Z}[i] = \{a + b\cdot i \mid a, b \in \mathbb{Z}\}$, where $i^2{=}-1$. Consider $\mathbb{Z}[i]$ as subset of complex numbers:

$$\mathbb{Z}[i] \subset \mathbb{C}$$

''The communication problem is solved'', announced Dr. Munchhausen to his fellow Peanolanders, ''because every number that we use in Peanoland has a corresponding number in Gaussland and we can use binary representations both to communicate and perform the conversion. Let me show you how ...''. Those were the last words of Dr. Munchhausen who was hit by a meteorite and never recovered.

Desperately, scientists studied his notebooks and found the following note:

> If $p \in \mathbb{N}$ and $g \in \mathbb{Z}[i]$, then p and g are equivalent if and only if, there exist $n \in \mathbb{N}$, $b_0 \in \{0, 1\}$, $b_1 \in \{0, 1\}$, $\ldots$, and $b_n \in \{0, 1\}$, such that
>
> $$p = \sum_{k=0}^{n} b_k \cdot 2^k \qquad \text{and} \qquad g = \sum_{k=0}^{n} b_k \cdot (i-1)^k$$

Another note said:

Eureka!
$292 = 20 - 6\cdot i$

Your task, to honor the memory of Dr. Munchhausen, is to build the translator that he intended to build in order to convert from the numbers used in Peanoland to the numbers used in Gaussland.

Input

The input consists of several test cases. Each test case consists of a line with a single natural number p in the Peanoland system ($0 \le p < 10^9$).

The input must be read from standard input.

Output

For each test case output two blank-separated integers a and b, where $g = a + b{\cdot}i$ is the number in the Gaussian system that corresponds to the natural number p in the Peanoland system.

The output must be written to standard output.

Sample Input	Sample Output

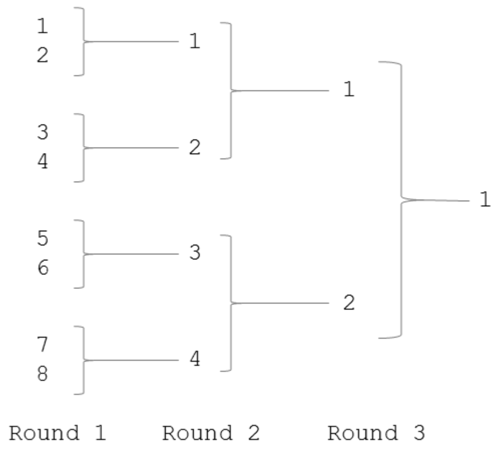
`0`
`1`
`2`
`3`
`4`
`292`
`999999999`
	`0 0`
	`1 0`
	`-1 1`
	`0 1`
	`0 -2`
	`20 -6`
	`5290 -5347`

C: Tennis Rounds

Source file name: `tennis.c`, `tennis.cpp`, `tennis.java`, *or* `tennis.py`
Author: Rodrigo Cardoso

In an N-rounds tennis tournament a group of 2^N players is seeded defining the first round to be played. Seeding means that each player is assigned a number between 1 and 2^N, and this assignment defines the round's draw as it establishes the matches that will be played among the players. In particular, the first round matches are numbered $1, 2, \ldots, 2^{N-1}$: match k will have player $2{\cdot}k-1$ vs. player $2{\cdot}k$, for $1 \le k \le 2^{N-1}$.

The winner of a match in the first round advances to the second round and the loser is eliminated. Consequently, the second round has exactly half the players of the first round. Moreover, if the winner of the first round match k would be reassigned the number k, then the second round's draw may be defined exactly as already explained for the first round. This assignment process could be repeated over and over again until there is exactly one player remaining, who happens to be the tournament's champion:



It is clear that the seeding process and the subsequent draws make it possible for any two players to eventually face each other in some round. For example, for N=3, players 2 and 5 could play at round 3 (the final), and players 5 and 7 could play at round 2 (one of the semifinals).

The tennis tournament organization is developing an online portal featuring many services. You have been hired to implement one of such services: given the seeding numbers of two players (i.e., their place in the first round ordering), the service should compute the round number in which these two players could eventually have a match against other.

Input

The input consists of several test cases, each one defined by a line containing three blank-separated integers N, i, and j, where N indicates the total number of rounds in the tournament ($1 \le N \le 20$), and i, j represent two seeding numbers at the first round ($1 \le i \le 2^N$, $1 \le j \le 2^N$, $i \ne j$).

The input must be read from standard input.

Output

For each test case, output a line with one integer indicating the round number in which players i and j may have a match in a tournament with N rounds.

The output must be written to standard output.

Sample input	**Output for the sample input**
`3 2 5`	`3`
`3 5 7`	`2`
`2 1 2`	`1`
`2 2 1`	`1`

D: **Tennis Championship**

Source file name: `champion.c, champion.cpp`, `champion.java`, *or* `champion.py`
Author: `Rafael García`

A certain tennis championship with P players has a particular set of rules:

1. Before every round, players are paired randomly.
2. Each pair so defined establishes a match that will be played.
3. The winner of a match advances to the next round in the tournament and the loser is eliminated from competition.
4. If the number of players before a round is odd, then one player (chosen at random) is automatically promoted to the next round.

This process should be repeated over and over again until there is exactly one player left. Such a player will be the champion.

The Tennis Championship Organization wants to calculate the total number of matches needed to determine the champion.

Input

The input consists of several test cases, each one consisting of a single line containing a positive integer P, the number of players.

The input must be read from standard input.

Output

For each test case, output a line with one integer indicating the number of matches needed to determine the champion.

The output must be written to standard output.

Sample Input	**Sample Output**
3 2	2 1