acm International Collegiate Programming Contest 2018 | icpc.foundation

ACM International Collegiate Programming Contest 2018

Latin American Regional Contests

November 9th-10th, 2018

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 6.

This problem set is used in simultaneous contests hosted in the following countries:

Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador
México, Panamá, Perú, República Dominicana and Venezuela

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.py2` or *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
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. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Test case is such that there are no ties when rounding as specified.

Problem A – Hours and Minutes

Author: Pablo Ariel Heiber, Argentina

Heidi has a discrete analog clock in the shape of a circle, as the one in the figure. Two hands rotate around the center of the circle, indicating hours and minutes. The clock has 60 marks placed around its perimeter, with the distance between consecutive marks being constant.
The minute hand moves from its current mark to the next exactly once every minute. The hour hand moves from its current mark to the next exactly once every 12 minutes, so it advances five marks each hour.
We consider that both hands move discretely and instantly, which means they are always positioned exactly over one of the marks and never in between marks.

At midnight both hands reach simultaneously the top mark, which indicates zero hours and zero minutes. After exactly 12 hours or 720 minutes, both hands reach the same position again, and this process is repeated over and over again. Note that when the minute hand moves, the hour hand may not move; however, when the hour hand moves, the minute hand also moves.

Heidi likes geometry, and she likes to measure the minimum angle between the two hands of the clock at different times of the day. She has been writing some measures down, but after several years and a long list, she noticed that some angles were repeated while some others never appeared. For instance, Heidi's list indicates that both at three o'clock and at nine o'clock the minimum angle between the two hands is 90 degrees, while an angle of 65 degrees does not appear in the list. Heidi decided to check, for any integer number A between 0 and 180, if there exists at least one time of the day such that the minimum angle between the two hands of the clock is exactly A degrees. Help her with a program that answers this question.

Input

The input consists of a single line that contains an integer A representing the angle to be checked ($0 \leq A \leq 180$).

Output

For each test case output a line containing a character. If there exists at least one time of the day such that the minimum angle between the two hands of the clock is exactly A degrees, then write the uppercase letter "`Y`". Otherwise write the uppercase letter "`N`".

Sample input 1	**Sample output 1**
90	Y

Sample input 2	**Sample output 2**
65	N

Sample input 3	**Sample output 3**
66	Y

Sample input 4	**Sample output 4**
67	N

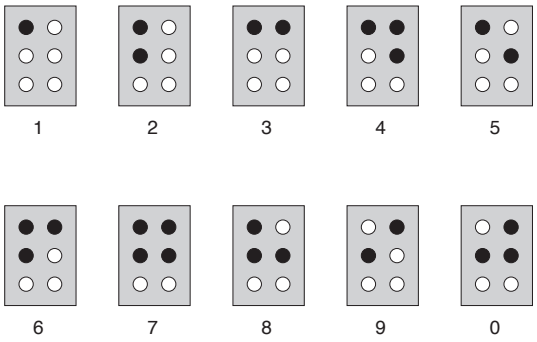
Sample input 5	**Sample output 5**
0	Y

This page would be intentionally left blank if we would not wish to inform about that.

Problem B – Braille

Author: Vinícius Santos, Brasil

The Braille system, designed by Louis Braille in 1825, revolutionized written communication for blind and visually impaired persons. Braille, a blind Frenchman, developed a tactile language where each element is represented by a cell with six dot positions, arranged in three rows and two columns. Each dot position can be raised or not, allowing for 64 different configurations which can be felt by trained fingers. The figure below shows the Braille representation for the decimal digits (a black dot indicates a raised position).



In order to develop a new software system to help teachers to deal with blind or visual impaired students, a Braille translator module is necessary. Given a braille message, composed only by braille digits, your job is to translate the message. Can you help?

Input

The first line contains an integer D indicating the number of digits in the braille message ($1 \leq D \leq 100$). The next three lines contain a message composed of D Braille cells that your program must translate. Braille cells are separated by single spaces. In each Braille cell a raised position is denoted by the character "`*`" (asterisk), while a not raised position is denoted by the character "`.`" (dot).

Output

For each test case print just the digits of the corresponding translation, in the same format as the input (see the examples for further clarification).

Sample input 1	**Sample output 1**
3 ** ** *. .* ** **	478

Sample input 2	**Sample output 2**
10 .* *. ** ** *. ** ** *. *. .* *. ** ** *. .* .* .. *. .. **	9876543210

Sample input 3	**Sample output 3**
3 .* .* .* ** ** **	000

This page would be intentionally left blank if we would not wish to inform about that.

Problem C – Help Cupid

Author: Pablo Ariel Heiber, Argentina

Cupid's job is getting harder, so he is adopting new technologies to help him with his difficult task of matching people into happy couples. He appointed the best programmers in his staff to a new project called Advanced Couples Matching (ACM). For this project, the programmers need to produce an algorithm that takes a set of an even number of N lonely persons and matches them into $N/2$ couples, such that each person is in exactly one couple.

Sadly, the data available about each person is limited. In this modern world, using gender, ethnicity, age or nationality as criteria to form couples is not a sensible option, so the programmers can only use data about the internet connection of each candidate. They decided to focus this stage on time zones. People living in closer time zones are more likely to find time to interact with each other. Thus, the programmers decided to create couples so as to minimize the *total time difference*.

Each time zone is identified by an integer between -11 and 12, inclusive, representing its difference in hours from a particular time zone called Coordinated Universal Time (or UTC). The time difference of two people living in time zones represented by integers i and j is the minimum between $|i - j|$ and $24 - |i - j|$. Given a partition of a set of an even number N of candidates into $N/2$ couples, its total time difference is the sum of the time difference of each couple.

You are asked to write a program that receives as input the time zones of a set of N candidates. The output of the program must be the minimum total time difference among all possible partitions of the set into couples.

Input

The first line contains an even integer N ($2 \leq N \leq 1000$) representing the number of candidates to be coupled. The second line contains N integers $T_1, T_2, \ldots, T_N$ ($-11 \leq T_i \leq 12$ for $i = 1, 2, \ldots, N$) indicating the time zones of the candidates.

Output

Output a single line with an integer the minimum total time difference among all possible partitions of the set of candidates into couples.

Sample input 1	**Sample output 1**
6 -3 -10 -5 11 4 4	5

Sample input 2	**Sample output 2**
2 -6 6	12

Sample input 3	**Sample output 3**
8 0 0 0 0 0 0 0 0	0

This page would be intentionally left blank if we would not wish to inform about that.