*We can all benefit by doing occasional "toy" programs, when artificial restrictions are set up, so that we are forced to push our abilities to the limit. …
The art of tackling miniproblems with all our energy will sharpen our talents for the real problems.*
Donald E. Knuth

Quarterfinal

Central region of Russia

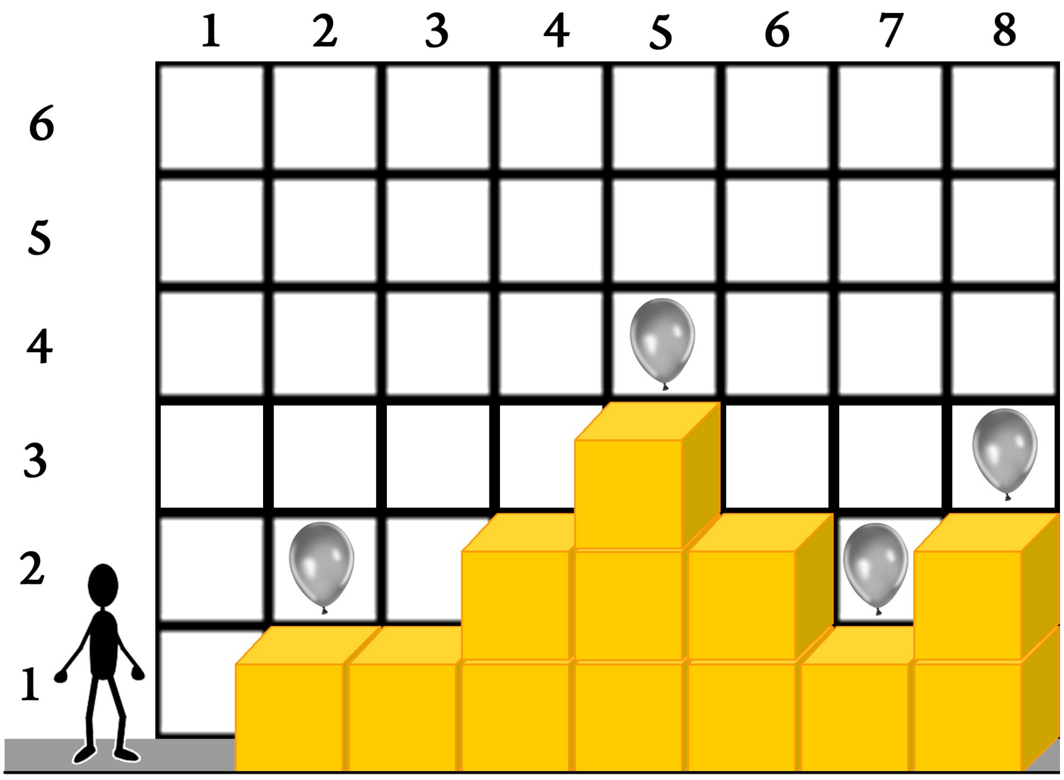
Rybinsk, October 15-16-2014

Input file name	INPUT.TXT
Output file name:	OUTPUT.TXT

A. Balloons (64 Mb, 1 sec / test)

A grid of shelves containing $\boldsymbol{k}$ balloons is attached to the wall. The position of each balloon is defined by integer coordinates: $\boldsymbol{x_i}$ is the column number in the shelf grid and $\boldsymbol{y_i}$ is the row number; $1 \le \boldsymbol{i} \le \boldsymbol{k}$. Student Vanya wants to gather all the balloons. In order to reach the balloons, he uses blocks that may be stacked on top of one another against the wall. Assume that the stock of blocks is unlimited.
Vanya can reach a balloon if he stands on a block that is located in the same column with the balloon, but one row below it. For example, if the balloon is in the sixth row it can be reached using a stack of five blocks.

Vanya attempts to place blocks against the wall so that he will be able to gather all the balloons going from left to right. On each step he can climb up or down at most 1 block. The starting position is the cell on level 0 before the wall.
Write a program to determine if it is possible to place blocks against the wall so that all the balloons will be reachable.



Limitations

k, x_i, y_i are integer numbers; $1 \le k \le 10{,}000$; $1 \le x_i, y_i \le 10{,}000$; $x_j < x_{j+1}$; $1 \le i \le k$; $1 \le j < k$.

Input

The first line of the input file contains an integer k, the number of balloons on the shelves. The following k lines define the positions of the balloons x_i, y_i (column numbers x_i are given in ascending order).

Output

The output file should contain a single word (without quotation marks): "YES" if it is possible to place blocks against the wall allowing to gather all the balloons, or "NO" otherwise.

Example

Input.txt	Output.txt

4
2 2
5 4
7 2
	YES

8 3	

2
2 2
3 5
	NO

1 1 2	YES
1 1 3	NO

B. Intellect Ltd (64 Mb, 1 sec / test)

Engineers working for *Intellect Ltd.* have proposed an innovative system to check the transfer of digital data. Only ***n***-digit decimal numbers divisible by 11 are used during transfer. The system was implemented on a remote space station transferring data to Earth. Eventually, the mission control center received a message containing (***n***+1) digits. The analysis indicated that the message had an extra spurious digit.
Write a program that will process the message and determine the number of ways to exclude a single digit so that the resulting number would be divisible by 11.

Limitations

$1 \le n \le 100\,000$.

Input

The first line of the input file defines the message length ***n***.
The second line contains a decimal number having ***n***+1 digits representing the message with an extra digit.

Output

The number of ways to exclude a single digit as described above.

Input.txt	Output.txt
3 1352	2
2 222	3

C. Race condition (64 Mb, 1 sec / test)

Vasya has written a program that launches ***n*** threads having m_i instructions each.
At any point in time the CPU is executing a single instruction from a single thread. The instructions in a thread are always executed in order (switching to other threads is possible).
After all instructions in a thread have been executed, the CPU ignores this thread.
Let us define an *execution path* as an ordered list of actually executed instructions from different threads.

Write a program to calculate the number of different execution paths (accounting for all possible switches between threads) for a multi-threaded program.

Limitations

$1 \le n \le 10$; $1 \le m_i \le 20$, $1 \le i \le n$, $\sum m_i \le 20$.

Input

The first line of the input file defines the number of threads n.
The second line contains n space-delimited integers m_i, the number of instructions in the threads.

Output

The number of different execution paths.

Input.txt	Output.txt
2 2 2	6
3 1 2 3	60

D. Ingress (64 Mb, 1 sec / test)

A new energy source has been discovered on the Earth, dubbed exotic matter, or XM. Many points of XM concentration were discovered. Such points are called portals. A faction was found, called the Enlightened, whose intent is to use XM to help an alien civilization, the Shapers, obtain ingress to our world. They can link portals to each other with links. A link is a straight line segment with endpoints at two portals. The links must not intersect, or the black hole will grow upemerge and swallow the galaxy. Every three portals linked to each other form a triangle called a control field. Control fields are an important part of the strategy aimed at setting control over the human civilization. All the people who are within a control field fall under control of the Enlightened. There is no relation between different control fields. A control field may share portals and links with another control field, or it may be located within another control field, or it may contain other control fields inside. There is another faction called the Resistance. Its members struggle against the intrusion of the Shapers.
They need your help in calculating the maximum number of control fields the Enlightened can make using a specified set of portals.

Limitations

$3 \le n \le 10{,}000$; $-300{,}000 \le x_i, y_i \le 300{,}000$

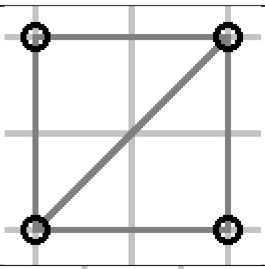
Input

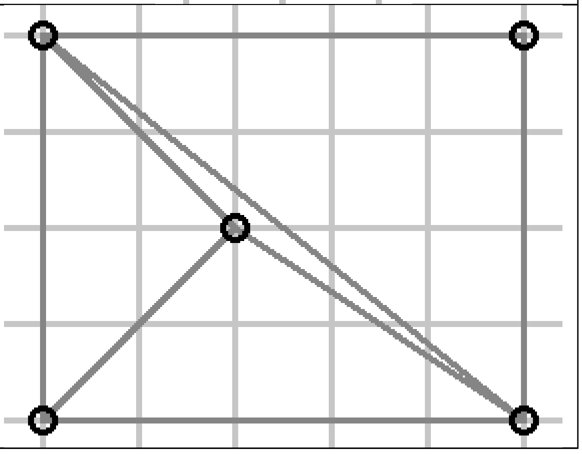
The first line contains an integer n, the number of portals. Each of the following n lines contains two space-delimited integers x and y, the coordinates of the i-th portal. It is guaranteed that any three portals form non-degenerate triangle.

Output

Write to the output file the maximum number of control fields that can be made.

Input	Output	Picture

4
0 0
2 0
2 2
0 2
	2	

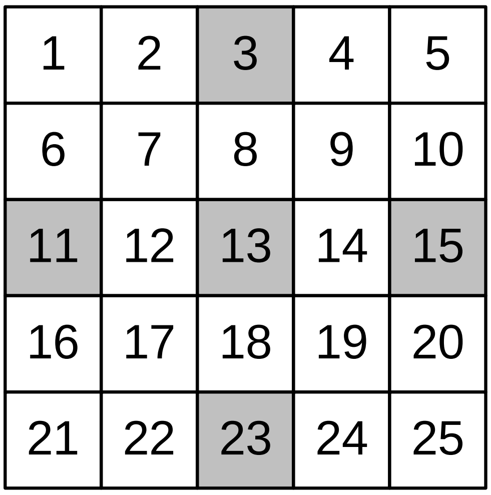
5
0 0
0 4
5 4
5 0
2 2
	5	

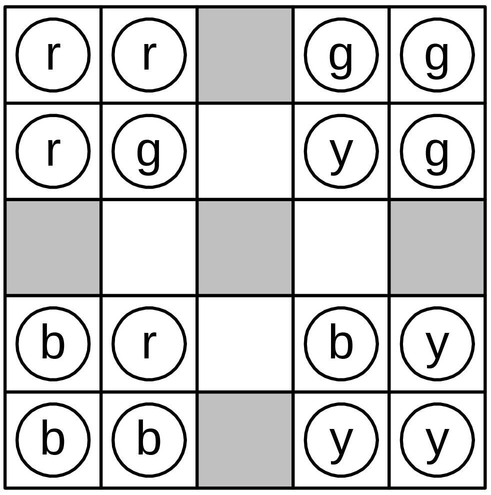
E. 4x4 (64 Mb, 1 sec / test)

The 4x4 game is played on a checkered board having 5 rows and 5 columns. The squares are numbered left to right, top to bottom. The squares with numbers 3, 11, 13, 15, and 23 are removed. So, the game board has four 2x2 corner blocks with bridges between them.

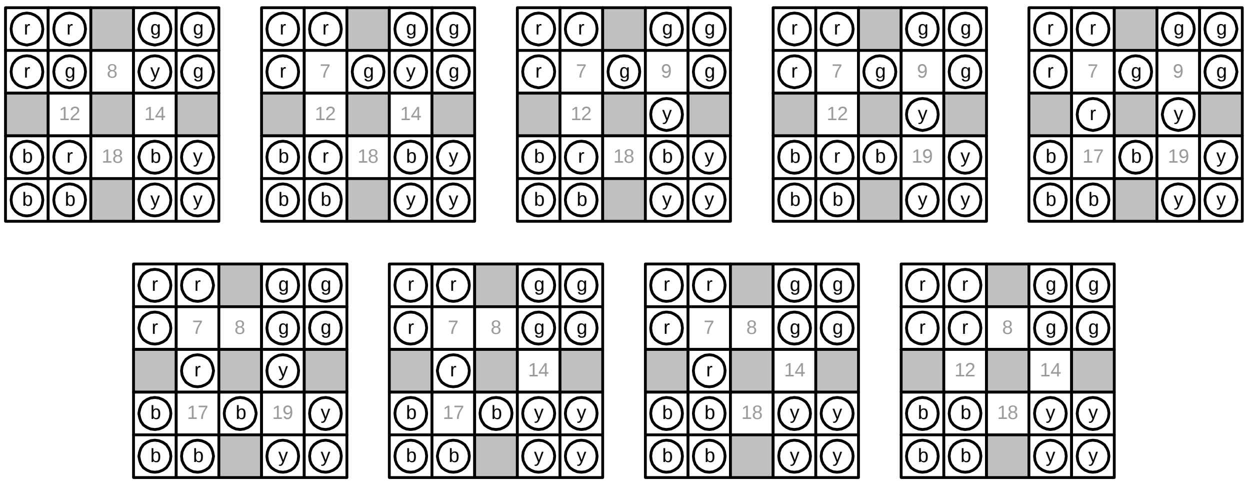
There are 16 game pieces on the field, four of each color: red, blue, yellow, and green. Initially, the pieces are placed in the corner blocks (i. e. the squares numbered 8, 12, 14, and 18 are empty).

The objective is to move the pieces so that all the pieces of each color were in the same corner block. A piece may only be moved to an empty square immediately to the right, to the left, above, or below. It is forbidden to move a piece to a square that is occupied by another piece, or to a removed square.





Your task is to write a program that, given the initial game position, will build a sequence of moves leading to the final position with pieces of the same color grouped in the corner blocks.



Note: such a sequence always exists, moreover, it is not unique. Any sequence of no more than 10,000 moves leading to the final position will be considered as a correct solution. The sequence does not have to be optimal.

Limitations

The resulting sequence may not be longer than 10 000 moves.

Input

The input file contains 5 lines of 5 characters each describing the initial position on the game board. The first line represents (left to right) the squares 1, 2, 3, 4, 5; the second line represents the squares 6, 7, 8, 9, 10 and so on. Period characters (“.”) stand for empty fields, “X” for removed fields, and “R”, “B”, “Y” and “G” for pieces of different colors.

Output

The output file should contain a sequence of moves resulting in the final position. Each move is represented on a separate line with two space-delimited integers: the number of the initial square and the number of the destination square.

Example

Input.txt	Output.txt

`RRXGG`
`RG.YG`
`X.X.X`
`BR.BY`
`BBXYY`
	`7 8`
	`9 14`
	`19 18`
	`17 12`
	`8 9`
	`14 19`
	`18 17`
	`12 7`

F. Sages (64 Mb, 1 sec / test)

A group of sages had a very lively discussion of sophisticated topics. After a while, feeling quite tired, they thought that some good tea and a big tasty apple pie would make them feel much better. However, the sages found it boring to cut the pie into equal pieces, so they took a more challenging path. The first sage takes $1/\boldsymbol{k}_1$-th part of the whole pie, the second $1/\boldsymbol{k}_2$-th, the third $1/\boldsymbol{k}_3$-th and so on. Here, the numbers $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, … $\boldsymbol{k}_n$ are different positive integers not exceeding 100 (even the wisest sage is unable to precisely cut off less than 1/100 of a pie).

Your task is to write a program that will split the pie in such a way that each sage gets $1/\boldsymbol{k}_i$-th of the whole pie (the pie must be shared without remainder!), or report that it is impossible to split the pie under these conditions. In case of several possible solutions, any of them will be considered correct.

Limitations

$1 \le \boldsymbol{n} \le 20$;

$\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n$ are positive integer numbers, $1 \le \boldsymbol{k}_i \le 100$;

$1/\boldsymbol{k}_1 + 1/\boldsymbol{k}_2 + 1/\boldsymbol{k}_3 + \ldots + 1/\boldsymbol{k}_n = 1$;

$\boldsymbol{k}_i \ne \boldsymbol{k}_j$, for any $\boldsymbol{i}, \boldsymbol{j}$, $\boldsymbol{i} \ne \boldsymbol{j}$.

Input

The input file contains a single integer $\boldsymbol{n}$, the number of sages sharing the pie.

Output

The output file should contain $\boldsymbol{n}$ space-delimited integers $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n$. If there is no solution then the output file should contain the message “No solution” (without quotation marks).

Example

Input.txt	Output.txt
`3`	`2 3 6`
`2`	`No solution`
`1`	`1`
`4`	`2 4 6 12`

G. Bus Conductor (64 Mb, 0.1 sec / test)

Vasya is a conductor on a bus serving quite an unpopular route, so he is loafing most of his working hours. In order to kill time he often plays a game. He takes a piece of his ticket roll and starts looking for lucky tickets. The tickets in the roll have sequential six-digit numbers. A ticket is lucky if the sum of the first three digits of its number is equal to the sum of the other three. Vasya records his observations making a string of capital Latin letters, writing down “L” for a lucky ticket, and “U” for a common one. For example, for the roll of 10 tickets numbered 001001, 001002, 001003, 001004, 001005, 001006, 001007, 001008, 001009, 001010, the resulting string will be “LUUUUUUUUL”.

Eventually, Vasya considered an inverse problem: would it be possible to come up with a ticket roll for a given string consisting of “L” and “U” characters? The answer turned out to be negative. For example, there is no roll having two consecutive lucky tickets, that is, no ticket roll exists for string “LL”.

Your task is to write a program that will find a ticket roll corresponding to the given string of “L” and “U” characters, or report that no such roll exists. In case of several possible solutions, the correct roll is the one with the least number of the first ticket.

Limitations

The input string contains capital letters “L” and “U” only. Its length does not exceed 1000 characters. The input string is non-empty.

Input

The first line of the input file contains an integer ***n***, the number of characters in the input string. The second line contains ***n*** characters. The only valid characters here are capital Latin letters “L” and “U”.

Output

The output file should contain either a six-digit integer, the number of the first ticket in the roll corresponding to the input string, or the message “No solution” (without quotation marks) if no such roll exists.

Example

Input.txt	Output.txt

`10`
`LUUUUUUUUU`
	`000000`

10 LUUULUUUUU	No solution
10 LUUUUUUUUL	001001

H. Cryptography (64 Mb, 1 sec / test)

One of the simplest encryption methods is to apply the bitwise exclusive or (**xor**) operator to the input message using an encryption key. However, this method is secure only when the input message is no longer that the encryption key, otherwise even a small portion of additional information about the original message might lead to a successful attack.

Suppose you are given an original message that is split into k 32-bit blocks. The blocks are encrypted independently by applying **xor** operator against the encryption key ***Key***, which is also 32 bits long. Each input block is a non-negative integer. It is known that the input blocks form a nondecreasing sequence $a_1, a_2, a_2, \ldots, a_k$.

You will be presented the encrypted sequence $b_1, b_2, b_3, \ldots, b_k$ ($b_i = a_i$ **xor** ***Key***). Your task is to unambiguously recover the encryption key ***Key***, or report that it is impossible to do so.

Limitations

$32 <= k <= 100\ 000$.

$0 <= a_i, b_i <= 2^{32} - 1$.

You can safely assume that the sequence $b_1, b_2, b_3, \ldots b_k$ is produced by applying **xor** operator to a nondecreasing sequence $a_1, a_2, a_3, \ldots a_k$ using some ***Key***.

Input

The first line of the input file contains an integer k. The second line contains k space-delimited integers $b_1, b_2, b_3, \ldots b_k$.

Output

The output file should contain the encryption key ***Key*** if it is possible to unambiguously recover it, or the word “Impossible” (without quotation marks) otherwise.

Example

Input.txt	Output.txt

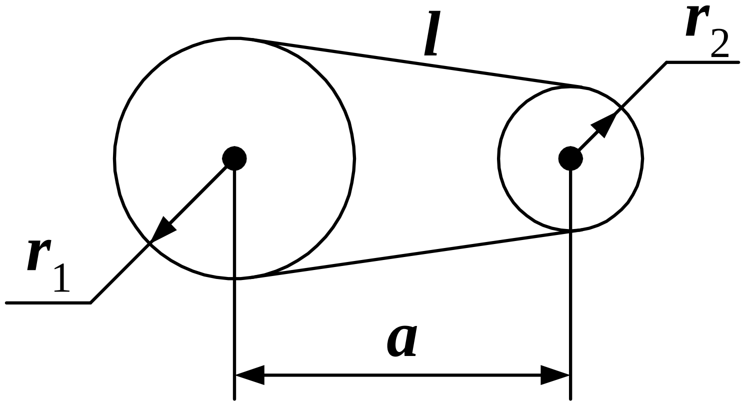
32
1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30
31 32
	Impossible

33
7 6 5 3 15 23 39 71 135 263
519 1031 2055 4103 8199
16391 32775 65543 131079
	7

262151 524295 1048583
2097159 4194311 8388615
16777223 33554439 67108871
134217735 268435463
536870919 1073741831
2147483655	

I. Belt Drive (64 Mb, 1 sec / test)

A belt drive consists of two pulleys of different radius ($\boldsymbol{r}_1$ and $\boldsymbol{r}_2$) connected by a belt of length ***l***. The industry manufactures belts of different standard lengths, so the engineers often face a task of calculating the distance between the pulley centers given their radii and the belt length. There are several simple approximate methods to do this, which are precise enough for rough estimations. However, some engineering tasks require more precise data.



Your task is to write a program that will calculate the distance ***a*** between the centers of the pulleys given their radii $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, and the length ***l*** of the belt.

Limitations

$1 \le \boldsymbol{r}_1, \boldsymbol{r}_2 \le 1{,}000$; $2\pi(\boldsymbol{r}_1 + \boldsymbol{r}_2) \le \boldsymbol{l} \le 10{,}000$.

Input

The first line contains 3 space-delimited real numbers precise to four decimal places: $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, ***l***. Trailing zeros following the decimal point may be omitted.

Output

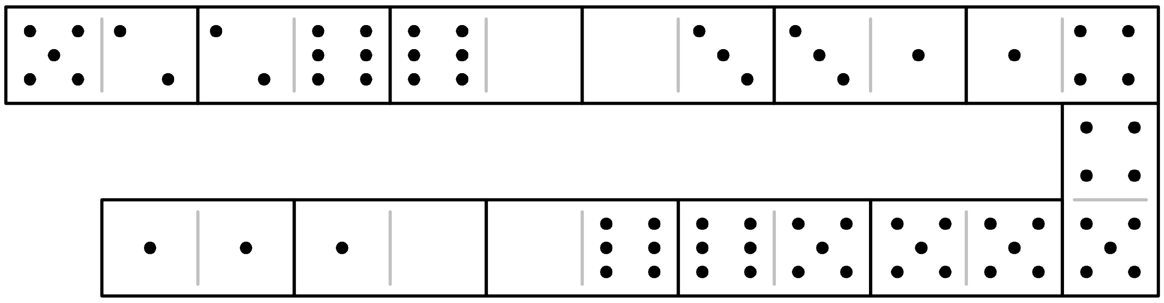
The output file should contain a single real number, the center-to-center distance ***a***. The answer must be precise to four decimal places.

Example

Input.txt	Output.txt
`1.0 2.0 20.0`	`5.191`
`1.0 2.0 18.85`	`4.6036`

J. Dominoes (64 Mb, 1 sec / test)

Vasya, the bus conductor who likes to look for lucky tickets, is also good at playing dominoes. One day someone brought him a number of domino tiles, and he wondered if it would be possible to use all the tiles making a valid line of play, meaning that adjacent tiles touch with an equal number of spots. See an example below (the line makes turns in the figure):



Note: each domino is a 1x2 rectangular tile consisting of two squares (ends). Each end is marked with up to 6 spots or is blank.
Your task is to write a program that determines if it is possible to make a continuous line of play using a given set of ***n*** tiles. Adjacent tiles in the line must have an equal number of spots on the touching ends.

Limitations

1 ≤ ***n*** ≤ 1000.

Input

The first line of the input file contains an integer ***n***, the number of tiles in the set. The following ***n*** lines describe the tiles. Each line contains two space-delimited integers from 0 to 6, the number of spots on the two ends of a tile.

Output

The program should output a single string (without quotation marks): "YES" if it is possible to make a continuous line of play using the given tiles, or "NO", otherwise.

Example

Input.txt	Output.txt

`2`
`1 2`
`1 3`
	`YES`

`3`
`1 2`
`1 3`
`1 4`
	`NO`

`2`
`1 2`
`3 4`
	`NO`

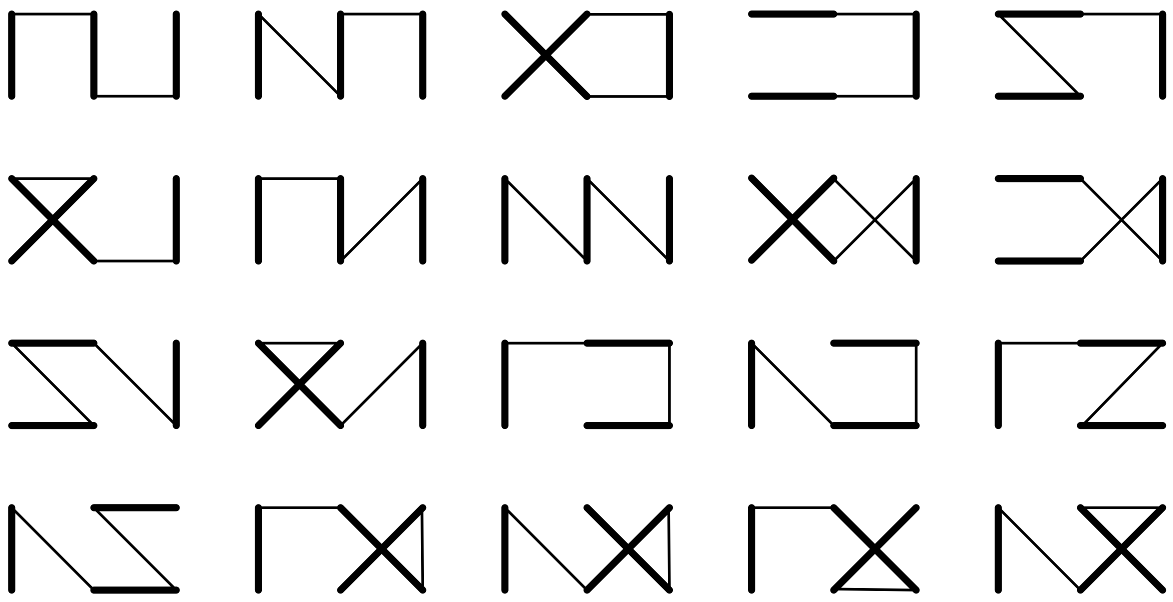
K. Embroidery (64 Mb, 1 sec / test)

A long band of fabric is about to be embroidered with a geometric pattern. The horizontally-oriented band is marked with a 1×***n*** guiding grid. The pattern is made according to the following rules.

1. The pattern starts in the lower left corner.
2. Every stitch connects two corners of a square in the guiding grid. The stitch may go in any direction: vertical, horizontal, or diagonal.
3. The fabric may be punctured only at the corners of grid squares, and only once for each corner. The final pattern must use all the corners.
4. The embroidery is done with a single continuous thread starting in the lower left corner of the grid and ending in any corner of any square.

Your task is to write a program that determines the number of different ways to embroider a band of size 1×***n***.

The figure below demonstrates all the 20 ways to embroider a 1×2 band according to the above rules. Front and back stitches are shown with lines of different thickness.



Limitations

1 ≤ ***n*** ≤ 30.

Input

The input file contains a single integer ***n***.

Output

The output file should contain a single integer, the number of possible designs.

Example

Input.txt	Output.txt
1	6

2	20
3	72

Task authors:

© Sergey G. Volchenkov, 2014	(volchenkov@yandex.ru)
© Vladimir N. Pinaev, 2014	(vpinaev@mail.ru)
© Michael Y. Kopachev, 2014	(rybkmu@mail.ru)
© Alexander Kiselyov, 2014	(diver.ru@gmail.com)
© Oleg Strekalovsky, 2014	(o.strekalovsky@yandex.ru)
© Victor Vinogradov, 2014	(fly@acm-server.ru)