Q1: Forbidden Number

Background

In the ancient kingdom of Arithma, there was a legend about a mystical number known as the Forbidden number.
One day, a young scholar named Elara stumbled upon an old manuscript in a dusty corner of the royal library.
The manuscript spoke of a challenge: "Take a number and break it into its digits. Raise each digit to the power of the number of digits, and if the sum equals the original number, you have discovered a Forbidden number."
Intrigued, Elara began her quest, testing various numbers and seeking the forbidden one.

Your task and explanations

You must help Elara in her journey to identify the forbidden numbers.

Input: 153
Output: TRUE
Explanation: $1^3 + 5^3 + 3^3 = 153$

Input: 1634
Output: TRUE
Explanation: $1^4 + 6^4 + 3^4 + 4^4 = 1634$

Q1: Forbidden Number

Test case:

#	Input	Output	Visibility
1	153	TRUE	visible
2	1634	TRUE	visible
3	371	TRUE	visible
4	372	FALSE	visible
5	1524	FALSE	visible
6	87963	FALSE	visible
7	457836	FALSE	visible
8	1343	FALSE	invisible
9	34567	FALSE	invisible
10	9474	TRUE	invisible

Q2: Tryst with Taylor

Given a real number x such that the absolute value of x is less than 1 (|x| < 1), and a positive integer k, we can calculate the value of log(1+x) using the first k terms of the Taylor series for the log function as follows:

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots + \frac{(-1)^{k+1}x^k}{k}$$

We will give you the real number x such that |x| < 1 and a positive integer k. You need to output the value of log(1+x) which was calculated using the first k terms. Give all outputs rounded to four decimal places. To print the result up to four decimal places you can use **"%.4lf"** assuming you need to use double data type.

Input: 0.1 5
Output: 0.0953

Note:

(1) The number x may be negative too, but the absolute value of it is less than 1.

(2) Use double variable to store x and use double typecasts and computations for this problem.

(3) Do not attempt to use the log() function from the math.h library to compute this. The math.h function uses a really large number of terms while calculating the log function whereas we will ask you to use smaller number of terms. You will fail the test cases if you try to use the math.h log function.

(4) Be careful about extra/missing lines and extra/missing spaces.

Q2: Tryst with Taylor

Test case:

#	**Input**	**Output**	**Visibility**
1	0.1 5	0.0953	visible
2	0.1 1	0.1000	visible
3	0.1 3	0.0953	visible
4	-0.1 1	-0.1000	visible
5	0.9 10	0.6262	visible
6	-0.9 7	-1.9870	visible
7	0 2	0.0000	visible
8	-0.753 7	-1.3576	invisible
9	0.53 8	0.4250	invisible

Q3: Ackermann

Given non-negative integers N and K, let T be the number of distinct, non-empty subarrays formed by the digits of N whose sum is at most K. Then, output T.

Notes:

- Given the decimal representation of a non-negative integer A with B digits as $a_1 a_2 \ldots a_B$, a subarray is a contiguous subsequence of the digits of A is any sequence of digits obtained from A of the form $a_i a_{i+1} \ldots a_j$, where $1 \leq i \leq j \leq B$.

- Two non-empty subarrays of A (following notation from the above point) $a_i a_{i+1} \ldots a_j$ and $a_k a_{k+1} \ldots a_m$ are considered distinct if $i \neq k$ or $j \neq m$ (or both).

Example:

Input: 1234 6

Output: 7

Explanation: We first list all the possible distinct, contiguous, non-empty subarrays, and their corresponding digit sums that can be formed by the digits of N=1234, as follows:

Q3: Ackermann….

Subarrays of 1234

S. No.	Subarray	Digit Sum
1	1	1
2	2	2
3	3	3
4	4	4
5	12	3
6	23	5
7	34	7
8	123	6
9	234	9
10	1234	10

Of these 10 subarrays, only 7 subarrays have their corresponding digit sum at most 6, i.e. subarrays 1, 2, 3, 4, 5, 6 and 8. Thus T=7.

Q3: Ackermann….

Test case:

#	Input	Output	Visibility
1	1234 6	7	visible
2	1111938529191 10	24	visible
3	1111111111111111 5	75	visible
4	0 0	0	visible
5	18 7	1	invisible
6	876978 98	21	invisible
7	23238 100	15	invisible
8	2 1	0	visible
9	93875190871098170 64	142	invisible

Q1: The Cab Driver

In a large city, a cab driver starts his daily journey from the origin at coordinates $(0,0)$. The driver receives several ride requests, each with a specified pickup and drop-off location. The driver must decide which rides to accept based on the following rules:

- The driver starts at location $(0,0)$ each day.
- For each ride, he picks up the passenger at (x_1, y_1) point and drops off at (x_2, y_2) point.
- After completing a ride and reaching the drop-off point (x_2, y_2), the driver will only accept the next ride if the pickup point of the next request is strictly within a 1-unit radius from his current location.
- If the pickup point is more than 1 unit away (i.e., >=1), the driver skips the ride and waits for the next request.

NOTE: The driver always goes to the pick-up location for the first trip.

Your task is to write a program that calculates the total distance covered by the driver for all accepted rides.

Input Format:

- The first line contains an integer, N, indicating the total number of rides requested.
- The next N lines contain four space separated integers x_1, y_1, x_2, y_2 representing the pickup and drop-off coordinates of each ride.

Output Format:

Q1: The Cab Driver…

Output Format:

Print a single number representing the total distance covered by the driver, rounded to two decimal places. Always use double precision variables in your code. No newline after printing the distance.

Constraints:

All coordinates are integers

Example

Input:

3
1 1 2 2
2 2 3 3
5 5 6 6

Output:

4.24

Q1: The Cab Driver…

Explanation:

The driver starts at (0,0) and drives to (1,1) to pick up the first passenger, then drives to (2,2) to drop them off.

$$Distance = \sqrt{(1-0)^2 + (1-0)^2} + \sqrt{(2-1)^2 + (2-1)^2} = 2.83$$

The next pickup at (2,2) is within 1 unit of the drop-off point (2,2), so the driver accepts the ride.

$$Distance = \sqrt{(3-2)^2 + (3-2)^2} = 1.41$$

The next pickup at (5,5) is more than 1 unit away from the drop-off point (3,3), so the driver skips this ride.

Total distance covered: $2.82 + 1.41 = 4.24$

Hint:

- Use the **sqrt** function while measuring the distance between two points. You can use the **sqrt** and **pow** functions from **math.h** library.
- Use the **countinue** keyword to skip an iteration of the loop.
- For each ride request, calculate the total distance from the current location to the pickup point and from the pickup point to the drop-off point. After completing each ride, check if the next ride's pickup point is within a 1-unit radius from your current drop-off point. If it is, continue to the next ride; otherwise, skip it.

Q1: The Cab Driver….

Test case:

#	Input	Output	Visibility
1	3 1 1 2 2 2 2 3 3 5 5 6 6	4.24	visible
2	3 2 2 3 3 5 5 6 6 7 7 8 8	4.24	invisible
3	1 1 1 2 2	2.83	invisible
4	3 1 1 2 2 5 5 6 6 2 2 3 3	4.24	visible
5	5 0 1 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9	2.00	invisible

Q1: The Cab Driver….

Test case:

#	Input	Output	Visibility
6	4 0 1 1 1 1 1 1 2 1 2 2 2 2 2 3 3	5.41	invisible
7	3 1 1 1 1 1 1 1 1 1 1 1 1	1.41	invisible
8	5 1 0 2 0 2 0 1 0 1 0 2 0 2 0 1 0 1 0 2 0	6.00	invisible
9	4 0 0 0 1 0 2 0 3 0 3 0 4 0 4 0 5	1.00	invisible

Q1: The Cab Driver….

Test case:

#	Input	Output	Visibility
10	5 1 1 2 2 3 3 4 4 2 2 3 3 5 5 6 6 4 4 5 5	4.24	invisible
11	3 10000 10000 9999 9999 9999 9999 10000 10000 10000 10000 10001 10001	14146.38	invisible
12	6 1 0 1 1 1 1 1 2 1 2 2 2 2 2 2 3 2 3 3 3 3 3 3 4	7.00	invisible

Q2: The Secret Code of the Ancient Vault

In a long-forgotten kingdom, there lies an ancient vault which contains unimaginable treasures. On the vault a number, N, is engraved. The vault can only be unlocked by deriving a secret code from N. To find the secret code, you must follow these steps:

Step 1: Start with the number N, which is engraved on the vault.

Step 2: For each number i from 1 to N, calculate the product of all the numbers from 1 to i (accumulating product).

Step 3: For each accumulated product:

- If the product is a number that can only be divided by 1 and itself, it's considered "pure".
- If the product ends in a 6 or 4, it's considered "special".
- If the product ends in 0, it's considered "neat".

Step 4: Sum all the "pure" products, "special" products, "neat" products you found in Step 3.

Step 5: Keep repeating the steps 1 to 4 after iteratively increasing the (current value of) N by 1, if any of the below two conditions are met:

- If the sum (from Step 4) is divisible by 4 and it does not end with two zeros.
- when the sum ends with two zeros, if the sum (from Step 4) is divisible by 4 before and after removing the two zeros.

Step 6: The final sum is the secret code to unlock the vault.

Q2: The Secret Code of the Ancient Vault

Your task is to write a C program that calculates the final sum, which is the secret code needed to unlock the vault, given N as input.

Input:

4

output:

33

Assume N is always a natural number. Use int variables only everywhere.

Hint: The accumulating product of 5 is: 1,2,6,24,120.

Q2: The Secret Code of the Ancient Vault

Test case:

#	Input	Output	Visibility
1	4	33	visible
2	7	5913	visible
3	5	153	invisible
4	9	409113	invisible
5	12	522956313	invisible
6	8	46233	invisible
7	2	3	visible
8	6	873	visible
9	3	9	invisible
10	10	4037913	invisible
11	400	363985433	invisible

Q1: Treasure Map Rotation

In the Kingdom of Arrays, a legendary treasure map has been discovered. The map consists of a sequence of ancient symbols arranged in a specific order, represented as an array of integers. The kingdom's sages have deciphered that to find the treasure; the map must be rotated a certain number of times to reveal the correct order of symbols.

Each day, the royal cartographer receives an integer k, which indicates how many steps to rotate the map to the right. To decipher the final map, you must write a program that performs the required rotations and displays the final sequence of symbols.

Input Format

- The first line contains two integers, n and k, where n is the number of symbols on the map and k is the number of steps the map should be rotated to the right.
- The second line contains n space-separated integers, a_1, a_2, ... a_n, representing the symbols on the map in their original order.

Output Format

- Print a single line containing the sequence of symbols after rotating the map to the right by k steps. Each number should be separated by a single space.

Q1: Treasure Map Rotation…

Explanation
The royal cartographer begins each day by rotating the map. A rotation by 1 step means the last symbol moves to the first position, and every other symbol shifts one position to the right.

Example

Input
7 3
1 2 3 4 5 6 7

Output
5 6 7 1 2 3 4

The original sequence is [1, 2, 3, 4, 5, 6, 7]; after rotating 3 steps to the right, the sequence becomes [5, 6, 7, 1, 2, 3, 4].

Assumption:
Assume that $1 < n < 10^5$

Q2: Missing Numbers

Given an array arr[ ] of size N-2 containing integers in the range 1 to N, where there are no duplicates, find the two missing numbers. Observe that the array will contain all numbers from 1 to N except for the two missing numbers.

Write a C program to find and print the two missing numbers given arr.

Input Format:

- The first line contains single integer N
- The second line contains N-2 space-separated integers in the range [1, N]

Output Format:

- Print a single line containing missing two numbers separated by a single space. There is newline after output. Of the two missing numbers, print the smaller number first and then print the larger number next.

Example

Input:

8
1 2 4 6 7 8

Output:

3 5

Q1: The Legend of Chess

There is a famous legend related to the origin of the game of chess. There was an emperor who was very impressed by this game and wished to reward the inventor. The conversation between them is given below. In this world however, the chessboard only has 7 squares on each side, for a total of 49 squares on the board.

Emperor: Name your reward, O wise one!

Inventor: O generous emperor, my wish is simple. I only wish for this -- give me one grain of rice for the first square of the chessboard, two grains for the next square, four for the next, eight for the next and so on for all 49 squares, with each square having double the number of grains as the square before.

We will provide you with the number of rice grains present with the emperor as an **unsigned long integer**

- If the king is able to fulfill his promise with those many rice grains, print YES in the first line and then in the next line, print how many grains of rice he will be left with after fulfilling his promise.
- If the king is not able to fulfill his promise with those many grains of rice, print NO in the first line and then in the next line, print the last square he was able to completely fill. This will be a number between 0 and 49.

Input:

2

Output:

NO

1

Q1: The Legend of Chess…

Note:

- The number of rice grains can be very large. Use an **unsigned long integer** type variable to do calculations.
- Be careful about extra/missing lines and extra/missing spaces.
- YOU MUST NOT use any library other than stdio. If you use, then you will be awarded zero marks for this question.

Test Cases:

#	Input	Output	Visibility
1	2	NO 1	visible
2	1125899906842624	YES 562949953421313	visible
3	9	NO 3	invisible
4	3	NO 2	invisible
5	0	NO 0	invisible

Q1: The Legend of Chess…

Test Cases:

#	Input	Output	Visibility
6	562949953421311	YES 0	invisible
7	562949953	NO 29	invisible
8	949953421311	NO 39	invisible
9	6294995342131	NO 42	invisible
10	56294995342131	NO 45	invisible
11	62949953421310	NO 45	invisible

Q1: The Legend of Chess…

Test Cases:

#	Input	Output	Visibility
12	6204995342131	NO 42	invisible
13	1125899906842624	YES 562949953421313	visible

Q2: Block Cipher

You will be given a long integer N and another integer K as input. You have to take the number N and divide it into blocks of length k. For example, if the number is 1999567890 and K = 3, then the blocks would be (1)(999)(567)(890). Note that the last block only has one digit.

Given this block structure, we derive an encrypted number as follows: take each block and add the last digit of that block to that block. Truncate this block to k digits i.e., preserve only the least significant k digits. For example, in the above case, we have k = 3, so

890 + 0 = (890)

567 + 7 = (574)

999 + 9 = 1(008)

1 + 1 = (2)

The encrypted number is formed by concatenating all the encrypted blocks i.e.

2008574890

On the first line, print the number of blocks in the integer. In the above example, there are 4 blocks in the number. On the second line print the encrypted number.

Q2: Block Cipher…

Note:

- Be careful about extra/missing lines and extra/missing spaces.
- Use unsigned long integer for processing and storing the plain and encrypted number.

EXAMPLE:

INPUT:

1999567890 3

OUTPUT:

4

2008574890

Q2: Block Cipher…

Test Cases:

#	Input	Output	Visibility
1	1999567890 3	4 2008574890	visible
2	123456 1	6 246802	visible
3	112233445566 2	6 122436486072	invisible
4	1234567890123456789 3	7 2238574890126462798	invisible
5	123456789 9	1 123456798	invisible
6	999999999999 6	2 8000008	invisible
7	1234567890123456789 7	3 1235067890143456798	invisible

Q2: Block Cipher…

Test Cases:

#	Input	Output	Visibility
8	12345690123456789 8	3 22345690223456798	invisible
9	123456390123456789 4	5 143462390223506798	invisible
10	1234563901234789 5	4 2234623901434798	invisible
11	999999991991 4	3 800081992	invisible
12	112233445566 5	3 122233845572	invisible

Q3: IRCTC Ticket

Your program must take as input age of a person (in years) and output the cost of ticket based on the following rules:

1. Age less than or equal to 2, then free (no cost).
2. Beyond 2 till 12 years (including 12), cost is Rs.490
3. Age 65 and above is 510
4. Everyone else it is 1000.

Input: take one number as input store it as unsigned short int.
Output: print cost of ticket (no new line).

Q3: IRCTC Ticket…

Test Cases:

#	Input	Output	Visibility
1	0	0	visible
2	15	1000	visible
3	6	490	visible
4	1	0	invisible
5	2	0	invisible
6	3	490	invisible
7	5	490	invisible
8	12	490	invisible
9	13	1000	invisible
10	50	1000	invisible
11	64	1000	invisible

Q3: IRCTC Ticket…

Test Cases:

#	Input	Output	Visibility
12	65	510	invisible
13	66	510	invisible
14	100	510	invisible

Q1: Number Play

By the Fundamental Theorem of Arithmetic, we know that any positive integer greater than unity has a unique prime factorization. Accordingly, let $N = \prod_{i=1}^{k} p_i^{e_i}$ be a unique prime factorization for N, where $p_{i+1} > p_i, \forall\, 1 \le i \le k-1$.

The task is to output the number S, where $S := \prod_{i=1}^{k} f(p_i)^{g(f(e_i+1))} \bmod M$. The functions $f: Z_{>1} \to Z_{\ge 1}$ and $g: Z_{>0} \to Z_{\ge 0}$ are defined as follows:

$$f(x) = \begin{cases} 2x+1, & if\ 2x+1\ is\ a\ prime \\ largest\ prime\ that\ devides\ x, & if\ 2x+1\ is\ NOT\ a\ prime \end{cases}$$

$$g(x) = \begin{cases} x^3 + 11 \bmod 3, & if\ x \bmod 4 < 2 \\ x^{19} - 8 \bmod 5, & if\ x \bmod 4 \ge 2 \end{cases}$$

Notes:

- $1 < N \le 1e6$
- $1 \le M \le 1e6$
- Use the template provided to code in your solution.

I/O format

Input:

N M

Output:

S

Q1: Number Play…

Example

Input:

600 20

Output:

7

Explanation:

- We first write the prime factorization of $N = 600$ is $600 = 2^3 * 3 * 5^2$
- Therefore, we need to output $S = f(2)^{g(f(4))} * f(3)^{g(f(2))} * f(5)^{g(f(3))} \; mod \; (20)$
- Accordingly, we first compute $f(2), f(3), f(4)$ and $f(5)$. Using the definition of $f(x)$, we have: $f(2) = 5$, $f(3) = 7$, $f(4) = 2$, $f(5) = 11$
- Next, we compute $g(f(4)) = g(2), g(f(2)) = g(5)$ and $g(f(3)) = g(7)$ as follows:
- $g(2) = 219 - 8 \; mod \; 5 = 0$, $g(5) = 53 + 11 \; mod \; 3 = 1$, $g(7) = 719 - 8 \; mod \; 5 = 0$
- Then $S = (50 * 71 * 110) \; mod \; (20) = 7$

Q2: Rhombus structure

Write a C program to display the number rhombus structure by defining a separate function to handle the printing of the rhombus. Your program should take an integer input n and then call a function to print the rhombus structure.

Requirements:

Define a function void printRhombus(int n) that takes an integer n as input and prints the rhombus structure.

In the main function, read the input n from the user and then call printRhombus(n) to display the output.

Input the number: 7

Expected Output :

```
      1
    212
  32123
 4321234
543212345
 4321234
  32123
    212
      1
```

Q1: Treasure Hunt

Imagine you're organizing a grand treasure hunt for a group of adventurers. After the hunt, each adventurer brings back a bag filled with different types of treasures—denoted as positive integer numbers between 1 to 100. As the event host, it's your job to list out every type of treasure that was found and how many of each type is in the bag.

Given an array (the bag) filled with treasures (the positive numbers) that may appear multiple times, your task is to create an inventory that prints out each type of treasure and its frequency of occurrence in the bag.

For example, if the bag contains: 10 4 4 1 10 10 4 2 10 1

Your inventory should show:

1 : 2
2 : 1
4 : 3
10 : 4

Help the adventurers keep track of their treasures with this organized list!

Input Format:

- The first line contains single positive integer N. Where N≤1000
- The second line contains N space-separated positive integers in the range [1, 100]

Q1: Treasure Hunt…

Output Format:

- If there are M unique treasure items, then output will have M lines.
- Every line will contain the treasure item and there corresponding frequency. Item and frequency should be separated by " : " (without quotes). Note that, before and after the colon, there is a single space.
- Every output line will be printed in increasing order based on the treasure item. Means, Print smallest number treasure item in first line, and highest number treasure item in the last line. See the example shown above.

Example:

INPUT:

10

10 4 4 1 10 10 4 2 10 1

OUTPUT:

1 : 2

2 : 1

4 : 3

10 : 4

Q2: Fixed Point Iteration

Given a function "f", which is a real-valued function over reals, your goal is to implement the so-called fixed point iteration algorithm that finds x such that $f(x) = x$.The algorithm is iterative in nature. It starts with an initial value x_0 (initial guess of fixed point solution). Then, at k^{th} iteration, it uses the formula:

$$x_{k+1} \equiv f(x_k).$$

Your program also needs to define the f function using the formula:

$$f(x) \equiv \frac{1}{2}\left(x + \frac{c}{x}\right), x > 0$$

where $c > 0$ is known.

Your program must take as input c, number of iterations, N, the algorithm must run for, initial value x_0. And must output x_N.

Input Format: c N x_0

e.g., 2 1000 0.1

Output Format: x_N (no newline). Print 10 digits after decimal point.

e.g., 1.4142135624

Note: Use **double** datatype for reals/rationals.

Q3: Jumbled match

Your program must read two words from user keyboard at runtime and determine whether one is a jumbled version of the other or not. For example, the words "jumble" and "bumlej" are jumbled versions of each other because there is one permutation of the letters in the word "jumble" that is same as the sequence of letters in "bumlej". Try writing a program that does not use nested loops. Assume the maximum number of letters in any string is 499 and the minimum number of letters is 1. Assume that words will only have the english alphabets and the matching is case-sensitive.

Input Format: two words separated by a single space

Output Format: print "yes" if they are jumbled versions of each other, else print "no". No newline.

No marks will be given if commands/functions that are NOT taught in lectures till now are used.

Input	**Output**
jumble bejulm	yes
jumble rumble	no
Jumble bejulm	no

Q4: Departmental Store

In a bustling marketplace, a merchant named Zara has a collection of N unique items, each represented by a number. Some items are more popular than the others, with customers buying them repeatedly; while the others get picked up just once or twice.

One day, Zara decides to organize her items based on their popularity. She plans to arrange them in such a way that the most popular items appear first, displayed prominently for her customers to see. However, Zara also has a quirky rule:

- if two items are equally popular, the smaller one (one with smaller item number) should always be placed first, showing off her fine taste in organization.

Can you help Zara sort her collection so that the most popular items shine at the top, and ties are broken by the item's number?

Input Format:

- The first line contains single positive integer N. Where N≤1000
- The second line contains N space-separated positive integers in the range [1, 100]

Output Format:

- Print all the items in a single line by decreasing order of their frequency. Each item should be separated by a single space.
- If two items are equally popular (same frequency), print smaller number item first followed by higher number order.

Q4: Departmental Store

Example:

INPUT:

11

10 4 4 1 10 10 4 2 1 10 1

OUTPUT:

10 10 10 10 1 1 1 4 4 4 2

Explanation:

The frequency of items 10, 4, 1, and 2 are 4, 3, 3, and 1, respectively. Since item 10 has the highest frequency (4), it should appear first in the array. Next, we have items 4 and 1, both with a frequency of 3. However, as 1 is smaller than 4, all occurrences of item 1 should come before item 4. Finally, item 2, with the lowest frequency of 1, will be placed at the end. Hence, the sorted array will begin with item 10, followed by item 1, item 4, and ending with item 2.

Q1: Frobenius Inner Product

You need to provide the definition of Frobenius inner-product between two matrices. The main function has already been provided to you. Please do not make any changes in it. Only fill in the function definition.

Input Format: first two numbers denote number of rows and columns of the matrices. Next few rows are for the first matrix, subsequent rows are for the second matrix. For e.g., If the matrices are

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and $B = \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$, then the input will be:

2 3

1 2 3

4 5 6

7 8 9

10 11 12

Output: the printf statement has already been written for you. Please do not change it or include additional printf statements. If your function definition is correct, then the correct value of inner product between the input matrices will be printed. For your convenience, the formula for inner-product is presented below.

$$\langle A, B \rangle \equiv \sum_{i,j} A_{ij} B_{ij}$$

Q2: The Tower of Magic

Story:

In a hidden valley, there stands an ancient Tower of Magic. Legends say that this tower holds untold wisdom, but it can only be accessed by solving a mystical challenge.

The challenge is simple in concept but complex in execution. At the top of the tower lives an old sorcerer who has a set of enchanted keys, each able to unlock a certain level of the tower. To climb the tower, you must determine the number of ways to reach the top, but there's a catch: you can only take 1 or 2 steps at a time.

The tower has N levels, and you are currently on the ground floor (level 0). Each step can take you up one or two levels, but no more. Your task is to find out how many distinct ways you can reach the top of the tower (level N).

Note: The old sorcerer has forbidden the use of arrays or strings to track your progress! You must rely on the pure magic of recursion to find the solution.

Input Format:

A single integer N , representing the number of levels in the tower.

Output

A single integer, the number of distinct ways to reach the top of the tower.

Q2: The Tower of Magic

Example

Input

3

Output

3

Explanation:

There are three ways to reach level 3:

1 -> 1 -> 1,

1 -> 2,

2 -> 1