CodeForge - B01 - Tính Tổng Hai Số (Method)

Độ khó: ★ Easy

Đề bài

Viết method `sum(int a, int b)` tính tổng hai số.

Input

- Hai số nguyên `a` và `b` trên hai dòng

Output

- In ra tổng `a + b`

Constraints

- `-10^9 ≤ a, b ≤ 10^9`

Ví dụ

Test case 1

Input:

```
5
3
```

Output:

```
8
```

Test case 2

Input:

```
-10
25
```

Output:

```
15
```

Test case 3

Input:

```
0
0
```

Output:

```
0
```

Tags: `method`, `basic`, `parameters`, `return-value`

CodeForge - B02 - Tìm Số Lớn Nhất (Method)

Độ khó: ★ Easy

Đề bài

Viết method `max(int a, int b, int c)` trả về số lớn nhất trong ba số.

Input

- Ba số nguyên `a`, `b`, `c` trên ba dòng

Output

- In ra số lớn nhất

Constraints

- `-10^9 ≤ a, b, c ≤ 10^9`

Ví dụ

Test case 1

Input:

```
5
3
8
```

Output:

```
8
```

Test case 2

Input:

```
-10
-5
-20
```

Output:

```
-5
```

Test case 3

Input:

```
100
100
99
```

Output:

```
100
```

Tags: `method`, `comparison`, `return-value`

CodeForge - B03 - Kiểm Tra Số Chẵn (Method)

Độ khó: ★ Easy

Đề bài

Viết method `isEven(int n)` kiểm tra số chẵn.

Trả về `true` nếu chẵn, `false` nếu lẻ.

Input

- Một số nguyên `n`

Output

- In ra `YES` nếu chẵn, `NO` nếu lẻ

Constraints

- `-10^9 ≤ n ≤ 10^9`

Ví dụ

Test case 1

Input:

```
4
```

Output:

```
YES
```

Test case 2

Input:

```
7
```

Output:

```
NO
```

Test case 3

Input:

```
0
```

Output:

```
YES
```

Tags: `method`, `boolean`, `modulo`

CodeForge - B04 - Tính Giai Thừa (Method)

Độ khó: ★ Easy

Đề bài

Viết method `factorial(int n)` tính giai thừa của n.

Input

- Một số nguyên không âm `n`

Output

- In ra n!

Constraints

- `0 ≤ n ≤ 20`

Ví dụ

Test case 1

Input:

```
5
```

Output:

```
120
```

Test case 2

Input:

```
0
```

Output:

```
1
```

Test case 3

Input:

```
10
```

Output:

```
3628800
```

Tags: `method`, `factorial`, `loop`

CodeForge - B05 - Kiểm Tra Số Nguyên Tố (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `isPrime(long n)` kiểm tra số nguyên tố.

Input

- Một số nguyên dương `n`

Output

- In ra `YES` nếu là số nguyên tố, `NO` nếu không

Constraints

- `1 ≤ n ≤ 10^12`

Ví dụ

Test case 1

Input:

```
7
```

Output:

```
YES
```

Test case 2

Input:

```
1
```

Output:

```
NO
```

Test case 3

Input:

```
100
```

Output:

```
NO
```

Test case 4

Input:

```
1000000007
```

Output:

```
YES
```

Tags: `method`, `prime`, `optimization`

CodeForge - B06 - Tính Lũy Thừa (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `power(long base, int exp)` tính base^exp.

Input

- Dòng 1: Số nguyên `base`
- Dòng 2: Số nguyên không âm `exp`

Output

- In ra base^exp

Constraints

- `0 ≤ base ≤ 100`
- `0 ≤ exp ≤ 18`
- Kết quả không vượt quá 10^18

Ví dụ

Test case 1

Input:

```
2
10
```

Output:

```
1024
```

Test case 2

Input:

```
5
0
```

Output:

```
1
```

Test case 3

Input:

```
3
15
```

Output:

```
14348907
```

Tags: `method`, `power`, `loop`

CodeForge - B07 - Tính GCD (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `gcd(long a, long b)` tính ước chung lớn nhất.

Sử dụng thuật toán Euclid.

Input

- Hai số nguyên dương `a` và `b` trên hai dòng

Output

- In ra GCD(a, b)

Constraints

- `1 ≤ a, b ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12
18
```

Output:

```
6
```

Test case 2

Input:

```
17
19
```

Output:

```
1
```

Test case 3

Input:

```
1000000000000
500000000000
```

Output:

```
500000000000
```

Tags: `method`, `gcd`, `recursion-or-loop`

CodeForge - B08 - Tính LCM (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `lcm(long a, long b)` tính bội chung nhỏ nhất.

Sử dụng công thức: LCM(a,b) = (a × b) / GCD(a,b)

Input

- Hai số nguyên dương `a` và `b` trên hai dòng

Output

- In ra LCM(a, b)

Constraints

- `1 ≤ a, b ≤ 10^9`

Ví dụ

Test case 1

Input:

```
12
18
```

Output:

```
36
```

Test case 2

Input:

```
5
7
```

Output:

```
35
```

Test case 3

Input:

```
1000000
1000000
```

Output:

```
1000000
```

Tags: `method`, `lcm`, `gcd`

CodeForge - B09 - Đếm Chữ Số (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `countDigits(long n)` đếm số chữ số của n.

Input

- Một số nguyên không âm `n`

Output

- In ra số lượng chữ số

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12345
```

Output:

```
5
```

Test case 2

Input:

```
0
```

Output:

```
1
```

Test case 3

Input:

```
1000000000000000000
```

Output:

```
19
```

Tags: `method`, `digit-count`, `loop`

CodeForge - B10 - Tổng Chữ Số (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `sumDigits(long n)` tính tổng các chữ số của n.

Input

- Một số nguyên không âm `n`

Output

- In ra tổng các chữ số

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12345
```

Output:

```
15
```

Test case 2

Input:

```
999
```

Output:

```
27
```

Test case 3

Input:

```
0
```

Output:

```
0
```

Tags: `method`, `digit-sum`, `loop`

CodeForge - B11 - Đảo Ngược Số (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `reverse(long n)` đảo ngược các chữ số của n.

Input

- Một số nguyên không âm `n`

Output

- In ra số sau khi đảo ngược

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12345
```

Output:

```
54321
```

Test case 2

Input:

```
1000
```

Output:

```
1
```

Test case 3

Input:

```
0
```

Output:

```
0
```

Tags: `method`, `reverse`, `digit-manipulation`

CodeForge - B12 - Kiểm Tra Palindrome (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `isPalindrome(long n)` kiểm tra số đối xứng.

Input

- Một số nguyên không âm `n`

Output

- In ra `YES` nếu là palindrome, `NO` nếu không

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12321
```

Output:

```
YES
```

Test case 2

Input:

```
12345
```

Output:

```
NO
```

Test case 3

Input:

```
0
```

Output:

```
YES
```

Tags: `method`, `palindrome`, `digit-manipulation`

CodeForge - B13 - Số Fibonacci (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `fibonacci(int n)` tính số Fibonacci thứ n.

F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)

Input

- Một số nguyên không âm `n`

Output

- In ra số Fibonacci thứ n

Constraints

- `0 ≤ n ≤ 90`

Ví dụ

Test case 1

Input:

```
0
```

Output:

```
0
```

Test case 2

Input:

```
10
```

Output:

```
55
```

Test case 3

Input:

```
50
```

Output:

```
12586269025
```

Tags: `method`, `fibonacci`, `loop`

CodeForge - B14 - Đếm Số Ước (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `countDivisors(long n)` đếm số lượng ước của n.

Input

- Một số nguyên dương `n`

Output

- In ra số lượng ước

Constraints

- `1 ≤ n ≤ 10^12`

Ví dụ

Test case 1

Input:

```
12
```

Output:

```
6
```

Test case 2

Input:

```
1
```

Output:

```
1
```

Test case 3

Input:

```
100
```

Output:

```
9
```

Tags: `method`, `divisors`, `optimization`

CodeForge - B15 - Tổng Các Ước (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `sumDivisors(long n)` tính tổng các ước của n.

Input

- Một số nguyên dương `n`

Output

- In ra tổng các ước

Constraints

- `1 ≤ n ≤ 10^12`

Ví dụ

Test case 1

Input:

```
12
```

Output:

```
28
```

Giải thích: 1 + 2 + 3 + 4 + 6 + 12 = 28

Test case 2

Input:

```
1
```

Output:

```

```

```
1
```

Test case 3

Input:

```
100
```

Output:

```
217
```

Tags: `method`, `divisors`, `sum`

CodeForge - B16 - Số Hoàn Hảo (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `isPerfect(long n)` kiểm tra số hoàn hảo.

Số hoàn hảo = tổng các ước thực sự (không bao gồm chính nó).

Input

- Một số nguyên dương `n`

Output

- In ra `YES` nếu là số hoàn hảo, `NO` nếu không

Constraints

- `1 ≤ n ≤ 10^9`

Ví dụ

Test case 1

Input:

```
6
```

Output:

```
YES
```

Test case 2

Input:

```
28
```

Output:

```
YES
```

Test case 3

Input:

```
100
```

Output:

```
NO
```

Tags: `method`, `perfect-number`, `divisors`

CodeForge - B17 - Số Armstrong (Method)

Độ khó: ★ ★ ★ Hard

Đề bài

Viết method `isArmstrong(long n)` kiểm tra số Armstrong.

Số Armstrong = tổng các chữ số lũy thừa bậc bằng số chữ số.

Ví dụ: 153 = $1^3 + 5^3 + 3^3$

Input

- Một số nguyên không âm `n`

Output

- In ra `YES` nếu là số Armstrong, `NO` nếu không

Constraints

- `0 ≤ n ≤ 10^9`

Ví dụ

Test case 1

Input:

```
153
```

Output:

```
YES
```

Test case 2

Input:

```
9474
```

Output:

```
YES
```

Test case 3

Input:

```
123
```

Output:

```
NO
```

Tags: `method`, `armstrong`, `digit-manipulation`

CodeForge - B18 - Method Overloading - Tính Diện Tích

Độ khó: ★★ Medium

Đề bài

Viết các method tính diện tích (overloading):

- `area(double r)`: Diện tích hình tròn ($\pi \times r^2$)
- `area(double a, double b)`: Diện tích hình chữ nhật ($a \times b$)
- `area(double a, double b, double h)`: Diện tích tam giác ($(a+b) \times h/2$)

Sử dụng $\pi = 3.14159$

Input

- Dòng 1: Số nguyên `type` (1, 2, hoặc 3)
- Các dòng tiếp theo: Các tham số tương ứng

Output

- In ra diện tích (làm tròn 2 chữ số thập phân)

Constraints

- Các giá trị > 0 và ≤ 1000

Ví dụ

Test case 1

Input:

```
1
5
```

Output:

```
78.54
```

Test case 2

Input:

```
2
10
5
```

Output:

```
50.00
```

Test case 3

Input:

```
3
8
12
5
```

Output:

```
50.00
```

Tags: `method-overloading`, `area`, `geometry`

CodeForge - B19 - Method Overloading - Tìm Max

Độ khó: ★★ Medium

Đề bài

Viết các method tìm max (overloading):

- `max(int a, int b)`: Max của 2 số
- `max(int a, int b, int c)`: Max của 3 số
- `max(int a, int b, int c, int d)`: Max của 4 số

Input

- Dòng 1: Số nguyên `n` (2, 3, hoặc 4)
- Dòng 2: n số nguyên cách nhau bởi dấu cách

Output

- In ra số lớn nhất

Constraints

- `-10^9 ≤ mỗi số ≤ 10^9`

Ví dụ

Test case 1

Input:

```
2
5 3
```

Output:

```
5
```

Test case 2

Input:

```
3
5 3 8
```

Output:

```
8
```

Test case 3

Input:

```
4
5 3 8 2
```

Output:

```
8
```

Tags: `method-overloading`, `max`, `comparison`

CodeForge - B20 - Số Chính Phương (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `isPerfectSquare(long n)` kiểm tra số chính phương.

Input

- Một số nguyên dương `n`

Output

- In ra `YES` nếu là số chính phương, `NO` nếu không

Constraints

- `1 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
16
```

Output:

```
YES
```

Test case 2

Input:

```
20
```

Output:

```
NO
```

Test case 3

Input:

```
1000000000000000000
```

Output:

```
YES
```

Tags: `method`, `perfect-square`, `math`

CodeForge - B21 - Chuyển Đổi Nhiệt Độ (Methods)

Độ khó: ★★ Medium

Đề bài

Viết các methods chuyển đổi nhiệt độ:

- `celsiusToFahrenheit(double c)`: C → F
- `fahrenheitToCelsius(double f)`: F → C

Input

- Dòng 1: Ký tự `type` ('C' hoặc 'F')
- Dòng 2: Nhiệt độ cần chuyển đổi

Output

- In ra nhiệt độ sau chuyển đổi (làm tròn 2 chữ số)

Constraints

- `-273.15 ≤ C ≤ 1000`
- `-459.67 ≤ F ≤ 2000`

Ví dụ

Test case 1

Input:

```
C
0
```

Output:

```
32.00
```

Test case 2

Input:

```
F
32
```

Output:

```
0.00
```

Test case 3

Input:

```
C
100
```

Output:

```
212.00
```

Tags: `method`, `conversion`, `temperature`

CodeForge - B22 - Tính Tổ Hợp C(n,k) (Method)

Độ khó: ★ ★ ★ Hard

Đề bài

Viết method `combination(int n, int k)` tính C(n,k).

Sử dụng công thức: C(n,k) = C(n-1,k-1) + C(n-1,k)

Input

- Dòng 1: Số nguyên không âm `n`
- Dòng 2: Số nguyên không âm `k`

Output

- In ra C(n,k)
- Nếu k > n, in ra 0

Constraints

- `0 ≤ n, k ≤ 60`

Ví dụ

Test case 1

Input:

```
5
2
```

Output:

```
10
```

Test case 2

Input:

```
10
5
```

Output:

```
252
```

Test case 3

Input:

```
60
30
```

Output:

```
118264581564861424
```

Tags: `method`, `combination`, `recursion-or-loop`

CodeForge - B23 - Tính Số Catalan (Method)

Độ khó: ★ ★ ★ Hard

Đề bài

Viết method `catalan(int n)` tính số Catalan thứ n.

C(0) = 1, C(n) = Σ C(i) × C(n-1-i) với i từ 0 đến n-1

Input

- Một số nguyên không âm `n`

Output

- In ra số Catalan thứ n

Constraints

- `0 ≤ n ≤ 30`

Ví dụ

Test case 1

Input:

```
0
```

Output:

```
1
```

Test case 2

Input:

```
5
```

Output:

```
42
```

Test case 3

Input:

```
10
```

Output:

```
16796
```

Test case 4

Input:

```
20
```

Output:

```
6564120420
```

Tags: `method`, `catalan`, `recursion`, `dp`

CodeForge - B24 - Tính Tổng Từ A Đến B (Method)

Độ khó: ★ Easy

Đề bài

Viết method `sumRange(int a, int b)` tính tổng từ a đến b.

◇ Input

- Dòng 1: Số nguyên `a`
- Dòng 2: Số nguyên `b` (a ≤ b)

◇ Output

- In ra tổng từ a đến b

◇ Constraints

- `1 ≤ a ≤ b ≤ 10^6`

Ví dụ

Test case 1

Input:

```
1
100
```

Output:

```
5050
```

Test case 2

Input:

```
5
10
```

Output:

```
45
```

Test case 3

Input:

```
100
100
```

Output:

```
100
```

Tags: `method`, `sum`, `range`

CodeForge - B25 - In Bảng Cửu Chương (Method)

Độ khó: ★ Easy

Đề bài

Viết method `printMultiplicationTable(int n)` in bảng cửu chương của n.

Input

- Một số nguyên dương `n`

Output

- In ra bảng cửu chương từ 1 đến 10
- Format: `n x i = kết quả`

Constraints

- `1 ≤ n ≤ 100`

Ví dụ

Test case 1

Input:

```
5
```

Output:

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Test case 2

Input:

```
3
```

Output:

```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

Tags: `method`, `void`, `multiplication-table`

CodeForge - B26 - Kiểm Tra Năm Nhuận (Method)

Độ khó: ★★ Medium

Đề bài

Viết method `isLeapYear(int year)` kiểm tra năm nhuận.

Quy tắc:

- Chia hết cho 4 VÀ không chia hết cho 100, HOẶC
- Chia hết cho 400

Input

- Một số nguyên dương `year`

Output

- In ra `YES` nếu là năm nhuận, `NO` nếu không

Constraints

- `1 ≤ year ≤ 10000`

Ví dụ

Test case 1

Input:

```
2024
```

Output:

```
YES
```

Test case 2

Input:

```
1900
```

Output:

```
NO
```

Test case 3

Input:

```
2000
```

Output:

```
YES
```

Tags: `method`, `leap-year`, `logic`

CodeForge - B27A - Giai Thừa (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy `factorialRecursive(int n)` tính giai thừa.

Input

- Một số nguyên không âm `n`

Output

- In ra n!

Constraints

- `0 ≤ n ≤ 20`

Ví dụ

Test case 1

Input:

```
5
```

Output:

```
120
```

Test case 2

Input:

```
0
```

Output:

```
1
```

Test case 3

Input:

```
15
```

Output:

```
1307674368000
```

Tags: `recursion`, `factorial`, `advanced`

CodeForge - B28A - Fibonacci (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy `fibonacciRecursive(int n)` tính số Fibonacci thứ n.

Lưu ý: Để tối ưu, có thể dùng memoization hoặc iterative approach.

Input

- Một số nguyên không âm `n`

Output

- In ra số Fibonacci thứ n

Constraints

- `0 ≤ n ≤ 45` (do recursion thuần không tối ưu)

Ví dụ

Test case 1

Input:

```
0
```

Output:

```
0
```

Test case 2

Input:

```
10
```

Output:

```
55
```

Test case 3

Input:

```
20
```

Output:

```
6765
```

Test case 4

Input:

```
40
```

Output:

```
102334155
```

Tags: `recursion`, `fibonacci`, `optimization`

CodeForge - B29A - GCD (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy `gcdRecursive(long a, long b)` tính GCD.

Sử dụng thuật toán Euclid đệ quy.

Input

- Hai số nguyên dương `a` và `b` trên hai dòng

Output

- In ra GCD(a, b)

Constraints

- `1 ≤ a, b ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12
18
```

Output:

```
6
```

Test case 2

Input:

```
1000000000000
500000000000
```

Output:

```
500000000000
```

Test case 3

Input:

```
17
19
```

Output:

```
1
```

Tags: `recursion`, `gcd`, `euclidean`

CodeForge - B30A - Lũy Thừa (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy `powerRecursive(long base, int exp)` tính base^exp.

Tối ưu hóa bằng phương pháp chia đôi (binary exponentiation).

Input

- Dòng 1: Số nguyên `base`
- Dòng 2: Số nguyên không âm `exp`

Output

- In ra base^exp

Constraints

- `0 ≤ base ≤ 100`
- `0 ≤ exp ≤ 18`

Ví dụ

Test case 1

Input:

```
2
10
```

Output:

```
1024
```

Test case 2

Input:

```
5
0
```

Output:

```
1
```

Test case 3

Input:

```
3
15
```

Output:

```
14348907
```

Tags: `recursion`, `power`, `binary-exponentiation`

CodeForge - B31A - Tổng Chữ Số (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy `sumDigitsRecursive(long n)` tính tổng các chữ số.

Input

- Một số nguyên không âm `n`

Output

- In ra tổng các chữ số

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12345
```

Output:

```
15
```

Test case 2

Input:

```
999
```

Output:

```
27
```

Test case 3

Input:

```
0
```

Output:

```
0
```

Tags: `recursion`, `digit-sum`, `advanced`

CodeForge - B32A - Đảo Ngược Số (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy để đảo ngược số.

Có thể dùng helper method với tham số bổ sung.

Input

- Một số nguyên không âm `n`

Output

- In ra số sau khi đảo ngược

Constraints

- `0 ≤ n ≤ 10^18`

Ví dụ

Test case 1

Input:

```
12345
```

Output:

```
54321
```

Test case 2

Input:

```
1000
```

Output:

```
1
```

Test case 3

Input:

```
0
```

Output:

```
0
```

Tags: `recursion`, `reverse`, `digit-manipulation`

CodeForge - B33A - Tháp Hà Nội

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy giải bài toán Tháp Hà Nội.

In ra các bước di chuyển đĩa từ cột A → C (qua cột B).

Input

- Một số nguyên dương `n` (số đĩa)

Output

- In ra các bước di chuyển
- Format: `Move disk [i] from [nguồn] to [đích]`

Constraints

- `1 ≤ n ≤ 10`

Ví dụ

Test case 1

Input:

```
3
```

Output:

```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

Test case 2

Input:

```
2
```

Output:

```
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
```

Tags: `recursion`, `tower-of-hanoi`, `classic-problem`

CodeForge - B34A - Tổ Hợp C(n,k) (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy tính C(n,k) sử dụng tam giác Pascal.

C(n,k) = C(n-1,k-1) + C(n-1,k)

Base cases: C(n,0) = C(n,n) = 1

Input

- Dòng 1: Số nguyên không âm `n`
- Dòng 2: Số nguyên không âm `k`

Output

- In ra C(n,k)

Constraints

- `0 ≤ n ≤ 30`
- `0 ≤ k ≤ n`

Ví dụ

Test case 1

Input:

```
5
2
```

Output:

```
10
```

Test case 2

Input:

```
10
5
```

Output:

```
252
```

Test case 3

Input:

```
20
10
```

Output:

```
184756
```

Tags: `recursion`, `combination`, `pascal-triangle`

CodeForge - B35A - In Số Nhị Phân (Recursion)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Viết method đệ quy chuyển số thập phân sang nhị phân.

Input

- Một số nguyên không âm `n`

Output

- In ra biểu diễn nhị phân của n

Constraints

- `0 ≤ n ≤ 10^9`

Ví dụ

Test case 1

Input:

```
10
```

Output:

```
1010
```

Test case 2

Input:

```
0
```

Output:

```
0
```

Test case 3

Input:

```
255
```

Output:

```
11111111
```

Test case 4

Input:

```
1024
```

Output:

```
10000000000
```

Tags: `recursion`, `binary`, `base-conversion`