Day 29 (14/07/2025)

---

1. Check if a Number is Even or Odd Using Bit Manipulation

A problem that introduces fundamental bitwise operations and teaches how to determine number parity using bit-level analysis instead of arithmetic operations.

You are given an integer **n**. Using **bitwise operations**, determine if the number is **even or odd**. Do not use the **modulus operator**. This problem demonstrates how bit manipulation can replace traditional arithmetic operations and is commonly used in **low-level programming** and **performance-critical applications** where bitwise operations are faster than division or modulus operations. The key insight is that the **least significant bit (LSB)** determines the parity of any number.

This teaches **bit-level number analysis** and **bitwise parity detection** techniques that are essential for **low-level programming and efficient arithmetic operations**.

Your task: Determine if a number is even or odd using only bitwise operations without arithmetic operators.

Examples

Input:

4

Output:

Even

Input:

7

Output:

Odd

---

2. Get the i-th Bit of a Number

A problem that demonstrates bit extraction techniques and teaches how to check specific bit positions using bitwise operations for binary data analysis.

Given an integer **n** and a position **i** (0-indexed from the right), find whether the **i-th bit is set (1) or not (0)** using bitwise operations. This operation is fundamental in **bit manipulation** and **binary data processing** where you need to **examine specific bit positions** for flags, permissions, or data encoding. The challenge involves understanding how to isolate and check individual bits using masking and shifting operations.

This introduces **bit masking** and **position-based bit extraction** techniques that are crucial for **binary data analysis and bit-level data processing**.

Your task: Extract and check the value of a specific bit position using efficient bitwise masking operations.

Examples

Input:

n = 5, i = 0

Output:

1

Input:

n = 5, i = 1

Output:

0

---

3. Set the i-th Bit of a Number

A problem that teaches bit modification techniques and demonstrates how to set specific bit positions using bitwise operations for binary data manipulation.

You are given an integer **n** and a position **i**. Set the **i-th bit** (0-indexed from right) of the number to **1** using bitwise operations and return the new number. This operation is commonly used in **system programming** and **flag management** where you need to **enable specific bits** for configuration settings, permissions, or status indicators. The challenge involves understanding how to modify individual bits without affecting other bit positions.

This teaches **bit setting** and **selective bit modification** techniques that are essential for **system programming and efficient bit-level data manipulation**.

Your task: Set a specific bit position to 1 using bitwise operations while preserving all other bits.

Examples

Input:

n = 5, i = 1

Output:

7

Input:

n = 8, i = 2

Output:

12