HackerRank Formatted Challenges

Challenge 1: Product Feature Configurator

Name: Product Feature Configurator

Description: Generate all possible feature combinations for a customizable product using recursive subset generation.

Problem Statement: You're developing a product configurator for a tech company that sells customizable laptops. Each laptop model has a set of available features (represented by feature IDs), and customers want to see all possible combinations they can choose from. Your task is to generate all possible subsets of features, including the empty set (no additional features).

For example, if a laptop model has features [1, 2, 3] where:

- 1 = Extended Warranty
- 2 = SSD Upgrade
- 3 = RAM Upgrade

The system should show all 8 possible combinations: no extras, individual features, pairs, and all three together.

Input Format:

- First line contains an integer n (1 ≤ n ≤ 10)
- Second line contains n space-separated integers representing feature IDs

Constraints:

- 1 ≤ n ≤ 10
- 1 ≤ feature_id ≤ 100
- All feature IDs are unique

Output Format: Print all possible subsets in lexicographic order. Each subset should be printed on a separate line with elements separated by spaces. Print "EMPTY" for the empty subset.

Tags: Recursion, Backtracking, Subsets, Product Configuration

Test Cases:

Test Case 1:

```
Input:
3
1 2 3

Output:
```

```
EMPTY
1
1 2
1 2 3
1 3
2
2 3
3
```

Test Case 2:

```
Input:
2
5 10

Output:
EMPTY
5
5 10
10
```

Test Case 3:

```
Input:
4
1 3 5 7

Output:
EMPTY
1
1 3
1 3 5
1 3 5 7
1 3 7
1 5
1 5 7
1 7
3
3 5
3 5 7
3 7
5
5 7
7
```

Test Case 4:

Input:
1
42

Output:
EMPTY
42

Test Case 5:

Input:
5
2 4 6 8 10

Output:
EMPTY
2
2 4
2 4 6
2 4 6 8
2 4 6 8 10
2 4 6 10
2 4 8
2 4 8 10
2 4 10
2 6
2 6 8
2 6 8 10
2 6 10
2 8
2 8 10
2 10
4
4 6
4 6 8
4 6 8 10
4 6 10
4 8
4 8 10
4 10
6
6 8
6 8 10
6 10
8
8 10
10

Challenge 2: System Performance Optimizer

Name: System Performance Optimizer

Description: Calculate optimization steps needed to reduce system load to zero using divide-and-conquer approach.

Problem Statement: You're working as a DevOps engineer for a cloud computing company. The system monitoring dashboard shows high CPU usage, and you need to implement an automated load balancer that reduces system load to zero. The load balancer follows a specific algorithm:

- If the current load is even, the system can distribute the load across 2 servers (divide by 2)
- If the current load is odd, the system must terminate one process first (subtract 1)

Your task is to calculate the minimum number of optimization steps needed to reduce the system load to zero using a recursive divide-and-conquer approach.

Input Format: A single integer num representing the current system load (1 ≤ num ≤ 10^6)

Constraints:

- 1 ≤ num ≤ 10^6

Output Format: Print a single integer representing the minimum number of steps required to reduce the load to zero.

Tags: Recursion, Divide and Conquer, System Optimization, Performance

Test Cases:

Test Case 1:

```
Input:
14

Output:
6
```

Test Case 2:

```
Input:
8

Output:
4
```

Test Case 3:

```
Input:
```

Input:
123

Output:
12

Test Case 4:

Input:
1

Output:
1

Test Case 5:

Input:
1000

Output:
13

Challenge 3: Robotic Warehouse Efficiency Calculator

Name: Robotic Warehouse Efficiency Calculator

Description: Calculate minimum moves needed for robotic arms to transfer container towers with size constraints using recursion.

Problem Statement: You're working as a systems analyst for an automated warehouse facility. The warehouse uses robotic arms to move container stacks between three positions: Source (A), Destination (C), and Buffer (B). Management needs to optimize operational efficiency by predicting the total number of moves required for any container transfer operation.

A robotic arm needs to move a tower of n containers from the Source to the Destination using the Buffer as an intermediate position. The robotic system has these constraints:

- Only one container can be moved at a time
- A larger container cannot be placed on top of a smaller container
- All containers start stacked on the Source position in descending order of size (largest at bottom)

Your task is to calculate the minimum number of moves required to transfer all containers from Source to Destination using a recursive divide-and-conquer approach. This helps the warehouse estimate operation time and resource allocation.

Input Format: A single integer n representing the number of containers ($1 \leq n \leq 20$)

Constraints:

- $1 \leq n \leq 20$

Output Format: Print a single integer representing the minimum number of moves required to transfer all containers from Source to Destination.

Tags: Recursion, Divide and Conquer, Tower of Hanoi, Robotics, Warehouse Management, Optimization

Test Cases:

Test Case 1:

```
Input:
1

Output:
1
```

Test Case 2:

```
Input:
2

Output:
3
```

Test Case 3:

```
Input:
3

Output:
7
```

Test Case 4:

```
Input:
4

Output:
15
```

Test Case 5:

```
Input:
10
```

Output:
1023