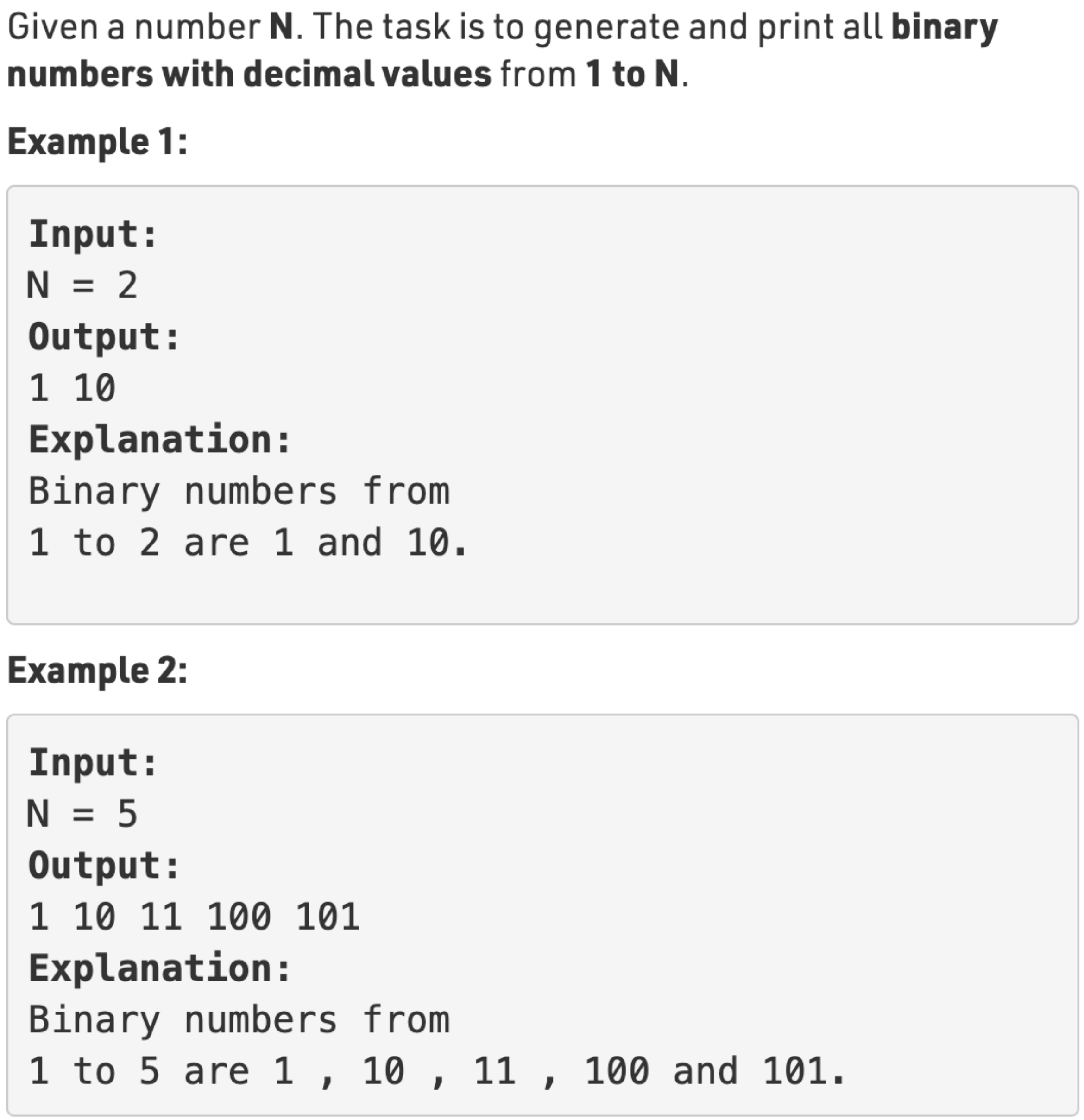
Queue:

1. Based on the concept of First In First Out (FIFO)
2. Real-life examples of stacks: any queue
3. Major operations in queues:
    1. **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
    2. **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
    3. **Front:** Get the front item from the queue.
    4. **size()**: returns the size of the queue
    5. **isEmpty**

Queue Problems:

Implement the push and pop functionality of a Queue using two Stacks.

Generate Binary Numbers



Given a number **N**. The task is to generate and print all **binary numbers with decimal values** from **1 to N**.

Example 1:

```
Input:
N = 2
Output:
1 10
Explanation:
Binary numbers from
1 to 2 are 1 and 10.
```

Example 2:

```
Input:
N = 5
Output:
1 10 11 100 101
Explanation:
Binary numbers from
1 to 5 are 1 , 10 , 11 , 100 and 101.
```

Linked List Problem:

Check if two Circular Linked Lists are identical.

Given two circular linked lists L1 and L2, the task is to find if the two circular linked lists are identical or not.

***Input**: L1: 1 -> 2 -> 3 -> 4 -> 5 -> 1 -> 2 -> 6*
L2: 5 -> 1 -> 2 -> 6 -> 1 -> 2 -> 3 -> 4
***Output**: Yes*
***Explanation:** If checked the 5th element of L1 and 1st element of L2 then they are identical.*
As they are circular, does not matter from where we start checking.

***Input**: L1: 1 -> 2 -> 3*
L2: 1 ->3 -> 2
***Output**: No*

Stack Problem:

Lexicographically largest possible String after removal of K characters.

Given a string S consisting of only lowercase letters, the task is to find the lexicographically largest string that can be obtained by removing K characters from the given string.

Input: *s = “zyxedcba”, K=1*
Output: *zyxedcb*
Explanation: *The character with the smallest ASCII value from the given string is ‘a’.*
Removal of ‘a’ generates the lexicographically largest possible string.

Input: *s = “abcde”, K=2*
Output: *cde*