Program 1: IPv4 Address Validation

1. Imagine you are working on a network configuration tool for a company.
This tool needs to validate the IPv4 addresses entered by users to ensure they are correctly formatted before saving them in the system.
An incorrectly formatted IP address could lead to network connectivity issues, making validation a crucial part of the tool.

Example Inputs and Outputs:
Input: "222.111.111.111"
Output: true

Input: "5555..555"
Output: false

Input: "0.0.0.255"
Output: true

Input: "0.0.0.0255"
Output: false

Program 1: Code

```
import java.util.Scanner;

public class IPValidator {
   public static boolean isValidIPv4(String ip) {
      String[] parts = ip.split("\."); // Splitting the IP using "."

      if (parts.length != 4) return false; // IPv4 must have 4 parts

      for (String part : parts) {
         if (part.isEmpty() || (part.length() > 1 && part.charAt(0) == '0')) return false; // No
empty parts & no leading zeros

         try {
            int num = Integer.parseInt(part); // Convert part to number
            if (num < 0 || num > 255) return false; // Check valid range
         } catch (Exception e) {
            return false; // If it's not a number, return false
         }
```

```
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an IPv4 address: ");
        String ip = sc.nextLine(); // Take input from user

        if (isValidIPv4(ip)) {
            System.out.println("Valid IPv4 Address!");
        } else {
            System.out.println("Invalid IPv4 Address!");
        }
        sc.close();
    }
}
```

Program 1: Output

Output for IPv4 Address Validation:

Input: "222.111.111.111"
Output: true

Input: "5555..555"
Output: false

Input: "0.0.0.255"
Output: true

Input: "0.0.0.0255"
Output: false

Program 2: Next Greater Element

2. You are building a stock price analysis tool.
One feature of this tool is to find the next greater price for each day's stock price in an array of daily stock prices.
Write a function that takes an array of daily stock prices as input and returns an array where each element is replaced by the next greater element to its right.
If there is no greater element, replace it with -1.

Example Inputs and Outputs:

Input: int[] arr = {6, 8, 0, 1, 3}
Output: [8, -1, 1, 3, -1]

Input: int[] arr = {1, 3, 2, 4}
Output: [3, 4, 4, -1]

Input: int[] arr = {10, 20, 30, 50}
Output: [20, 30, 50, -1]

Input: int[] arr = {50, 40, 30, 10}
Output: [-1, -1, -1, -1]

Program 2: Code

```
import java.util.Arrays;
import java.util.Scanner;

public class NextGreaterElement {
  public static int findNext(int[] arr, int index, int start) {
    if (start >= arr.length) return -1; // No greater element found
    if (arr[start] > arr[index]) return arr[start]; // Found next greater element
    return findNext(arr, index, start + 1); // Keep checking next elements
  }

  public static void fillNextGreater(int[] arr, int index, int[] result) {
    if (index >= arr.length) return; // Base case for recursion
    result[index] = findNext(arr, index, index + 1); // Find next greater element
    fillNextGreater(arr, index + 1, result); // Move to next index
  }

  public static int[] nextGreater(int[] arr) {
    int[] result = new int[arr.length]; // Create result array
    fillNextGreater(arr, 0, result);
```

```
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter array size: ");
        int size = sc.nextInt();
        int[] arr = new int[size];

        System.out.println("Enter array elements: ");
        for (int i = 0; i < size; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("Next Greater Elements: " + Arrays.toString(nextGreater(arr)));
        sc.close();
    }
}
```

Program 2: Output

Output for Next Greater Element:

Input: {6, 8, 0, 1, 3}
Output: [8, -1, 1, 3, -1]

Input: {1, 3, 2, 4}
Output: [3, 4, 4, -1]

Input: {10, 20, 30, 50}
Output: [20, 30, 50, -1]

Input: {50, 40, 30, 10}
Output: [-1, -1, -1, -1]