Week 15- Day 2 : Coding Challenge

(Maximum marks -15)

Q-1) Validate IP Address

https://leetcode.com/problems/validate-ip-address/

(5 marks)

(Medium)

Given a string `IP`, return `"IPv4"` if IP is a valid IPv4 address, `"IPv6"` if IP is a valid IPv6 address or `"Neither"` if IP is not a correct IP of any type.

A valid IPv4 address is an IP in the form `"x1.x2.x3.x4"` where `0 <= xi <= 255` and `xi` cannot contain leading zeros. For example, `"192.168.1.1"` and `"192.168.1.0"` are valid IPv4 addresses but `"192.168.01.1"`, while `"192.168.1.00"` and `"192.168@1.1"` are invalid IPv4 addresses.

A valid IPv6 address is an IP in the form `"x1:x2:x3:x4:x5:x6:x7:x8"` where:

- `1 <= xi.length <= 4`
- `xi` is a hexadecimal string which may contain digits, lower-case English letter (`'a'` to `'f'`) and upper-case English letters (`'A'` to `'F'`).
- Leading zeros are allowed in `xi`.

For example, `"2001:0db8:85a3:0000:0000:8a2e:0370:7334"` and `"2001:db8:85a3:0:0:8A2E:0370:7334"` are valid IPv6 addresses, while `"2001:0db8:85a3::8A2E:037j:7334"` and `"02001:0db8:85a3:0000:0000:8a2e:0370:7334"` are invalid IPv6 addresses.

Example 1:

Input: IP = "172.16.254.1"
Output: "IPv4"
Explanation: This is a valid IPv4 address, return "IPv4".

Q-2)Letter Combinations of a Phone Number (5 marks)
https://leetcode.com/problems/letter-combinations-of-a-phone-number/
(Medium)
Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]

Example 2:

Input: digits = ""
Output: []

Q-3) Unique Paths III (5 marks)
https://leetcode.com/problems/unique-paths-iii/
(Hard)

On a 2-dimensional grid, there are 4 types of squares:

- 1 represents the starting square. There is exactly one starting square.
- 2 represents the ending square. There is exactly one ending square.
- 0 represents empty squares we can walk over.
- -1 represents obstacles that we cannot walk over.

Return the number of 4-directional walks from the starting square to the ending square, that walk over every non-obstacle square exactly once.

Example 1:

Input: [[1,0,0,0],[0,0,0,0],[0,0,2,-1]]
Output: 2
Explanation: We have the following two paths:
1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2)
2. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2)

Example 2:

Input: [[1,0,0,0],[0,0,0,0],[0,0,0,2]]
Output: 4
Explanation: We have the following four paths:
1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2),(2,3)
2. (0,0),(0,1),(1,1),(1,0),(2,0),(2,1),(2,2),(1,2),(0,2),(0,3),(1,3),(2,3)
3. (0,0),(1,0),(2,0),(2,1),(2,2),(1,2),(1,1),(0,1),(0,2),(0,3),(1,3),(2,3)
4. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2),(2,3)

Marks distribution:
Question 1,2 and 3 carry 5 marks each.