13. Roman to Integer
This question is very simple. We only need to check two characters exist in SpecialKeys(two characters), if so, we will skip by 2, or we will skip by one. TC is O(1)

```
class Solution:
    def romanToInt(self, s: str) -> int:
        Table = {'M': 1000, 'D': 500, 'C': 100, 'L': 50, 'X': 10, 'V': 5, 'I': 1, 'IX': 9, 'IV': 4, 'XL': 40, 'XC':
90, 'CD':400, 'CM': 900}
        SpecialKey = set(['IX', 'IV', 'XL', 'XC', 'CD', 'CM'])
        result = 0
        length = len(s)
        idx = 0

        while idx < length:
            if idx + 1 < length and s[idx:idx+2] in SpecialKey:
                result += Table[s[idx:idx+2]]
                idx += 2
            else:
                result += Table[s[idx]]
                idx += 1
        return result
```

929. Unique Email Addresses
We just follow the rules extract useful address from the original email address, then add it to our set. In the end, we only need to count set's size. TC is O(mn)

```
class Solution:
    def numUniqueEmails(self, emails: List[str]) -> int:
        email_address = set()

        for email in emails:
            local_name, host_name = email.split('@')
            email_address.add(''.join(local_name.split('+')[0].split('.')) + '@' + host_name)

        return len(email_address)
```

344. Reverse String
Nothing to say

```
class Solution:
    def reverseString(self, s: List[str]) -> None:
        """
        Do not return anything, modify s in-place instead.
        """
        length = len(s) - 1
        half = len(s) // 2

        for i in range(half):
            s[i], s[length - i] = s[length - i], s[i]
```

1137. N-th Tribonacci Number
Nothing to say.

```
class Solution:
    def tribonacci(self, n: int) -> int:
        t1, t2, t3 = 0, 1, 1
        result = 0

        if n == 0:
            return 0
        if n == 1:
            return 1
        if n == 2:
            return 1
        for i in range(n - 2):
            result = t1 + t2 + t3
            t1, t2, t3 = t2, t3, result
        return result
```

1138. Alphabet Board Path
This question is quite tricky. We will use map to record each character’s coordinate. Then we could calculate delta x and y according to the difference of destination and source. But for ‘z’, when destination is ‘z’, we need to move horizontally first, than vertically, in other cases, we move vertically firstly and horizontally next. TC is O(n).

```
class Solution:
    def alphabetBoardPath(self, target: str) -> str:
        memo = {}
        board = ["abcde", "fghij", "klmno", "pqrst", "uvwxy", "z"]
        rows = len(board)
        result = ''
        x, y = 0, 0

        for i in range(rows):
            for j in range(len(board[i])):
                memo[board[i][j]] = (i, j)
```

```
for c in target:
    t_x, t_y = memo[c]
    d_y = t_y - y
    d_x = t_x - x
    if c == 'z':
        if d_y > 0:
            result += "R" * d_y
        else:
            result += "L" * (-d_y)

        if d_x > 0:
            result += "D" * d_x
        else:
            result += "U" * (-d_x)
    else:
        if d_x > 0:
            result += "D" * d_x
        else:
            result += "U" * (-d_x)
        if d_y > 0:
            result += "R" * d_y
        else:
            result += "L" * (-d_y)

    result += '!'
    x, y = t_x, t_y
return result
```