DAY 3

INTERVIEW BIT PROBLEMS :

1. First Missing Positive Integer

Given an unsorted integer array, find the first missing positive integer.

Example:

Given [1,2,0] return 3,

[3,4,-1,1] return 2,

[-8, -7, -6] returns 1

Your algorithm should run in **O(n)** time

CODE :

PYTHON

Using O(n) extra space:

```
class Solution:
    # @param A : list of integers
    # @return an integer
    def firstMissingPositive(self, A):
        maxi=max(A)
        if maxi<1:
            return 1
        if len(A)==1:
            if A[0]==1:
                return 2
            else:
                return 1
        out_list=[0]*maxi
        for i in range(len(A)):
            if A[i]>0:
```

```
                    if out_list[A[i]-1]!=1:
                        out_list[A[i]-1]=1

            for i in range(len(out_list)):
                if out_list[i]==0:
                    return i+1
            return i+2
```

C++

```
int Solution::firstMissingPositive(vector<int> &A) {
    int n = A.size();
    for (int i = 0; i < n; i++) {
        if (A[i] > 0 && A[i] <= n) {
            int pos = A[i] - 1;
            if (A[pos] != A[i]) {
                swap(A[pos], A[i]);
                i--;
            }
        }
    }
    for (int i = 0; i < n; i++) {
        if (A[i] != i + 1) return (i + 1);
        }
        return n + 1;
    }
```

2. Rotate Matrix

You are given an n x n 2D matrix representing an image.
Rotate the image by 90 degrees (clockwise).
You need to do this in place.
Note that if you end up using an additional array, you will only receive partial score.
Example:

If the array is

```
[
    [1, 2],
    [3, 4]
]
```

Then the rotated array becomes:

```
[
    [3, 1],
    [4, 2]
]
```

CODE :

PYTHON

```
class Solution:
    # @param A : list of list of integers
    # @return the same list modified
    def rotate(self, A):
        n=len(A[0])
        for i in range(n//2):
            for j in range(i,n-i-1):
                temp=A[i][j]
                A[i][j]=A[n-j-1][i]
                A[n-1-j][i]=A[n-1-i][n-1-j]
                A[n-1-i][n-1-j] = A[j][n-1-i]
                A[j][n-1-i] = temp
        return A
```

C++

```
void Solution::rotate(vector<vector<int> > &A) {
    // Do not write main() function.
    // Do not read input, instead use the arguments to the function.
```

```
    // Do not print the output, instead return values as specified
    int n=A.size();
    if(n==1||n==0){
        return ;
    }
    for(int i=0;i<n/2;i++){
        for(int j=i;j<(n-i-1);j++){
            int temp=A[i][j];
            A[i][j]=A[n-j-1][i];
            A[n-j-1][i]=A[n-i-1][n-j-1];
            A[n-i-1][n-j-1]=A[j][n-i-1];
            A[j][n-i-1]=temp;
        }
    }
};
```

3. Find Permutation

Given a positive integer n and a string s consisting only of letters D or I, you have to find any permutation of first n positive integer that satisfy the given input string.
D means the next number is smaller, while I means the next number is greater.

Note:
Length of given string s will always equal to n - 1
Your solution should run in linear time and space.

Example :

Input 1:

n = 3

s = ID

Return: [1, 3, 2]

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @param B : integer
    # @return a list of integers
    def findPerm(self, A, B):
        if B==0:return[]

        elif B==1: return [1]
        c=A.count('D')
        output =list()
        cd=c
        ci=c+1
        output.append(ci)
        ci+=1
        for i in range(B-1):
            if A[i]=='D':
                output.append(cd)
                cd-=1
            else :
                output.append(ci)
                ci+=1
        return output
```

C++

```
vector<int> Solution::findPerm(const string A, int B) {
    vector<int>output;
    int cd=0,ci=0;
    int n=A.size();
    for(int i=0;i<n;i++){
        if(A[i]=='D'){
```

```
            cd+=1;
        }
    }
    ci=cd+1;
    output.push_back(ci);
    ci+=1;
    for(int i=0;i<n;i++){
        if(A[i]=='I'){
            output.push_back(ci);
            ci+=1;
        }
        else{
            output.push_back(cd);
            cd-=1;
        }
    }
    return output;
}
```