Array Extra class

Q.No 1

i/p → {23, -7, 12, -10, -11, 40, 60}

modify ⇒ -ve → Left.
+ve → Right

Approach ⇒

1. Sorting
   T.C ⇒ O(nlogn)
2. Sort of 1
   2 pointer Approach
3. temp Array
   O(n) space.
   O(n) Time

j (isme hum -ve rakh sakte hai)

| 23 | -7 | 12 | -10 | -11 | 40 | 60 |

arr

index
(For Traversing)

for arr[index] > 0,
we ignore
for arr[index] < 0
we swap with j index element

Code:

```
void shiftNegativeOneside (int arr[],
    int n) {
    int j = 0;
  // j → memory block → jaha pr
  // main -ve;
  // number store. Kar sakta hu.
  for (int index = 0; index < n; index++) {
  // index → entire array ko traverse
  // karne k liye
    if (arr[index] < 0) {
      swap(arr[index], arr[j]);
        j++;
    }
  }
}
```

Q.2 Sort colors

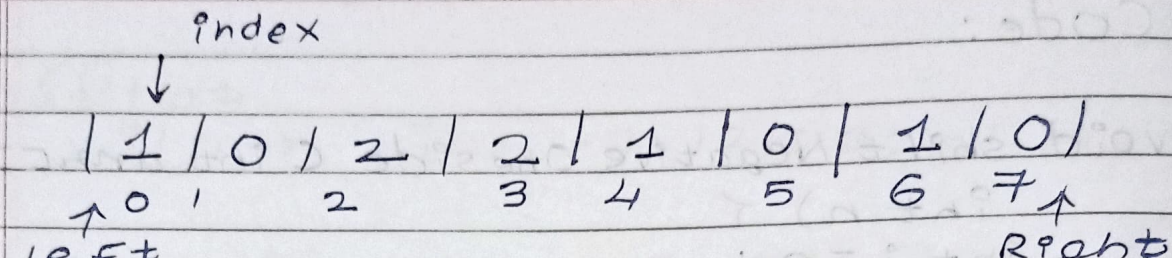
in-place → No extra space.

i/p ⇒ |0|1|1|2|0|2|1|

o/p ⇒ |0|0|1|1|1|2|2|

Approach ⇒

1. counting
2. sorting
3. 2-pointer Approach.



Logic ⇒

1. 0 mila toh left ko dedunga
2. 2 mila toh right ko dedunga
3. 1 mila ignore

Code:

```
void sortColors(vector<int>& nums){
    int n = nums.size();
    int index = 0;
    int left = 0;
    int right = n-1;

  while(index <= right){
    if(nums[index] == 0){
        swap(nums[index], nums[left]);
        left++;
        index++;
    }
   else if(nums[index] == 2){
   swap(nums[index], nums[right]);
   right--; //Catch ⇒ no need of index++
   }
   else{
    index++;
   }
  }
}
```

Rotate Array

ip ⇒
10 | 20 | 30 | 40 | 50 | 60 $k = 2$

op ⇒
60 | 10 | 20 | 30 | 40 | 50

50 | 60 | 10 | 20 | 30 | 40

Approach :-

1. Modulus of k
2. Temp Array.

If $k = 2$

ip (Index)	op (Index)
0	→ 2
2	→ 4
1	→ 3
3	→ 5
4	→ 0
5	→ 1

↳ (index + k) % n

Code:

```
void rotate (vector<int>& nums, int k){
    int n= nums.size();
    vector<int>ans(n);

for(int index=0; index<n ; index++){
    int newIndex =(index+k)%n ;
    ans[newIndex]=nums[index];
}
nums = ans;
}
```

How can we do in O(1) ?
If we do step by step rotating.

T.C → O(n)
S.C → O(n)

Missing Number

Approach

1. for( )
   ↳ search → Linear search
   T.C → O(N2)

2. sort → nlogn
   Traverse → difference = 1

3. Finding sum

Rearrange Array Elements

A.P. ⇒ sum = $\frac{n}{2}(a+l)$

n ← No. of terms; a ← 1st term; l ← last term

Code:

```
int missingNumber(vector<int>& nums){
    int sum=0;
    int n = nums.size();
    for(int index=0; index<n; index++){
        sum = sum+ nums[index];
    }
    int totalSum = ((n)*(n+1)/2);
    int ans = totalSum -sum;
    return ans;
}
```

Time complexity → O(n)

Row with Maximum ones.

	0	1	2	3	
0	1	0	0	0	→ 1
1	0	1	1	0	→ 2
2	0	1	1	0	→ 2
3	0	1	1	0	→ 3
4	0	0	1	0	→ 1

We have to maintain {rowNo, count}

code:

```
vector<int> rowAndMaximumOnes (vector
<int> vector<int>& mat) {
    vector<int> ans;
    int n = mat.size();
  // oneCount → will store max ones
   int oneCount = INT_MAX;
  // RowNO → will store index of
// max No. of 1's wali row.
   int rowNO = -1;
  for(int i=0; i<n; i++) { int count
    for(int j=0; j<mat[i].size()j++
      if(mat[i][j]==1) {
        count++;
      }
  // after row completion, compare
// count with oneCount.
  if(count>oneCount) {
      oneCount = count;
      rowNO = i;
    }
}

ans.pushback(rowNO);
 ans.pushback(oneCount);
 return ans;
}
```

Time complexity → O(N*m)

Rotate Image (VVVV IMP)

1	2	3
4	5	6
7	8	9

$\Rightarrow$

7	4	1
8	5	2
9	6	3

Transpose

1	4	7
2	5	8
3	6	9

Arr → Transpose → Array]
90 degree

Code :

```
void rotate ( vector<vector<int>>& matrix){
    int n = matrix.size();
  // Transpose
  for( int i=0; i<n; i++) {
    for( int j = i ; j< matrix[i].size();j++) {
    swap (matrix[i][j], matrix [j] [i]) ;
}}
// Reverse → 2D Matrix ki saari rows ko
// kithi rows h → 0 > (n-1)
for(int i=0 ; i<n ; i++) {
    // hr row ko reverse krna h
  reverse ( matrix[i].begin(), matrix[i].
                                end()) );
}}
};
```