DAY 2

1) Rotate Matrix/image 90*:

```
class Solution {
public:
    void rotate(vector<vector<int>>& m) {
        for(int i=0;i<m.size();i++){
            for(int j=0;j<i;j++){
                swap(m[i][j], m[j][i]);
            }
        }
        for(int i=0;i<m.size();i++){
            reverse(m[i].begin(), m[i].end());
        }
    }
};
```

2) Merge Overlapping subintervals

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& itr) {
        vector<vector<int>> ans;
        sort(itr.begin(), itr.end());
        ans.push_back(itr[0]);
        for (int i = 1; i < itr.size(); i++) {
            if (itr[i][0] <= ans.back()[1]) {
                ans.back()[1] = max(itr[i][1], ans.back()[1]);
            } else {
                ans.push_back(itr[i]);
            }
        }
        return ans;
    }
};
```

3) Merge two sorted arrays without extra space.

```

class Solution {
public:
    void merge(long long arr1[], long long arr2[], int n, int m) {
        int i = n - 1;
        int j = 0;
        while (i >= 0 && j < m) {
            if (arr1[i] >= arr2[j]) {
                swap(arr1[i], arr2[j]);
                i--;
                j++;
            } else {
                break;
            }
        }
        sort(arr1, arr1 + n);
        sort(arr2, arr2 + m);
    }
};
```

GAP Method by Varun sir:

```
int nextGap(int gap) { return (gap <= 1) ? 0 : ((gap / 2) + (gap % 2)); }
void mergeArrays(vector<int>& a, vector<int>& b) {
    int n = a.size(), m = b.size();
    int gap = (m + n) / 2;
    while (gap > 0) {
        int i = 0, j = gap;
        while (j < n + m) {
            int& val1 = (i < n) ? a[i] : b[i - n];
            int& val2 = (j < n) ? a[j] : b[j - n];
            if (val1 > val2) {
                swap(val1, val2);
            }
            i++;
            j++;
        }
        gap = nextGap(gap);
    }
}
```

4) Find a Duplicate in an array of N+1 integers. (slow-fast approach)

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int slow = 0;
        int fast = 0;
        do {
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while (slow != fast);
        slow = 0;
        while (slow != fast) {
            slow = nums[slow];
            fast = nums[fast];
        }
        return slow;
    }
};
```