**Day-12**

**Problem1**- [**378. Kth Smallest Element in a Sorted Matrix**](https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/)Solution-

class Solution {

public:

int kthSmallest(vector<vector<int>>& m, int k) {

int n=m.size();

int z=0;

vector<int>v;

for(auto &i:m)

{

for(int j=0;j<i.size();j++)

{

v.push\_back(i[j]);

}

}

sort(v.begin(),v.end());

return v[k-1];

}

};

**Problem2**- [**767. Reorganize String**](https://leetcode.com/problems/reorganize-string/)

Solution-

class Solution {

public:

string reorganizeString(string s) {

int n=s.size();

priority\_queue<pair<int,char>>pq;

map<char,int>m;

for(auto i:s)

m[i]++;

for(int i=0;i<n;i++)

{

if(m[s[i]]>(n+1)/2)

return "";

}

string result="";

for(auto i:m)

{

pq.push({i.second,i.first});

}

while(pq.size()>=2)

{

auto p=pq.top();

pq.pop();

auto q=pq.top();

pq.pop();

result+=p.second;

p.first--;

result+=q.second;

q.first--;

if(p.first>0)pq.push(p);

if(q.first>0)pq.push(q);

}

if(!pq.empty())

result+=pq.top().second;

return result;

}

};

**Problem 3**- [**239. Sliding Window Maximum**](https://leetcode.com/problems/sliding-window-maximum/)

class Solution {

public:

vector<int> maxSlidingWindow(vector<int>& nums, int k) {

int n=nums.size();

deque<int>dq;

vector<int>v;

for(int i=0;i<n;i++)

{

if(!dq.empty()&& dq.front()==i-k)

dq.pop\_front();

while(!dq.empty() && nums[dq.back()]<nums[i])

dq.pop\_back();

dq.push\_back(i);

if(i>= k-1)

v.push\_back(nums[dq.front()]);

}

return v;

}

};

**Problem 4**- [**234. Palindrome Linked List**](https://leetcode.com/problems/palindrome-linked-list/)

class Solution {

public:

bool isPalindrome(ListNode\* head) {

vector<int>v;

while(head!=NULL)

{

v.push\_back(head->val);

head=head->next;

}

int n=v.size();

cout<<n;

int flag=0;

for(int i=0;i<n/2;i++)

{

if(v[i]!=v[n-i-1])

return false;

}

return true;

}

};

**Problem 5**- [**70. Climbing Stairs**](https://leetcode.com/problems/climbing-stairs/)

class Solution {

public:

int solve(int n, vector<int>&dp)

{

if(n<=2)

return n;

if(dp[n]!=-1)return dp[n];

return dp[n]= solve(n-1,dp)+solve(n-2,dp);

}

int climbStairs(int n) {

// return climbStairs(n-1)+climbStairs(n-2);

vector<int>dp(n+1,-1);

return solve(n,dp);

}

};