Assignment : 05

1. 2007. Find Original Array From Doubled Array

An integer array original is transformed into a doubled array changed by appending twice the value of every element in original, and then randomly shuffling the resulting array.

Given an array changed, return original if changed is a doubled array. If changed is not a doubled array, return an empty array. The elements in original may be returned in any order.

```
//Time Complexity: O(n)

//Space Complexity: O(n)

class Solution {

public:

    vector<int> findOriginalArray(vector<int>& changed) {

        int n=changed.size();

        vector<int>ans;

        map<int,int>mp;

        if(n%2==0)

        {

            for(auto x:changed)

            {

                mp[x]++;

            }

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

            for(auto x:changed)

            {

                if(mp[x]==0)
                {
                    continue;
                }
                if(mp[2*x]==0)
                {
                    return {};
                }
                if(mp[x]&&mp[2*x])
                {
                    mp[2*x]--;
                    ans.push_back(x);
                    mp[x]--;
                }
            }
        }
        return ans;

    }
};
```

2. Find Minimum in Rotated Sorted Array

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

- [4,5,6,7,0,1,2] if it was rotated 4 times.
- [0,1,2,4,5,6,7] if it was rotated 7 times.

```
//Time Complexity: O(log n)

// Space Complexity: O(1)

class Solution {

public:

    int findMin(vector<int>& nums) {

        if(nums.size()==0)

        {

            return -1;

        }

        if(nums.size()==1)

        {

            return nums[0];

        }

        int left=0;

        int right=nums.size()-1;

        while(left<right)

        {

            int mid=left+(right-left)/2;

           if(mid>0 && nums[mid]<nums[mid-1])
```

```
                {
                    return nums[mid];
                }
                else if(nums[mid]>=nums[left] && nums[mid]>nums[right])
                {
                    left=mid+1;
                }
                else
                {
                    right=mid-1;
                }
            }
            return nums[left];
        }
    };
```

3. Convert 1D Array Into 2D Array

You are given a 0-indexed 1-dimensional (1D) integer array original, and two integers, m and n. You are tasked with creating a 2-dimensional (2D) array with m rows and n columns using all the elements from original.

```
//Time Complexity: O(m*n)

//Space Complexity: O(m*n)

class Solution {

public:

    vector<vector<int>> construct2DArray(vector<int>& original, int m,
int n) {

        int N=original.size();

        vector<vector<int>>ans(m,vector<int>(n));

        if(N!=(m*n))

        {

            return {};

        }

        else{

            int x=0;



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

            {

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

                {

                    ans[i][j]=original[x++];
                }

            }


        }

         return ans;


    }

};
```