ASSIGNMENT-5

1.

```
class Solution {
 public:
  vector<vector<int>> construct2DArray(vector<int>& original, int m, int n) {
    if (original.size() != m * n)
      return {};

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

    for (int i = 0; i < original.size(); ++i)
      ans[i / n][i % n] = original[i];

    return ans;
  }
};
```

2.

```
class Solution {
public:
int arrangeCoins(int n) {
int ans = 0;
while((ans+1)*1LL*(ans+2)<=(long long)n*2){
ans++;
}
return ans;
}
};
```

3.

```
#include <bits/stdc++.h>
using namespace std;

// Function to sort an square array
void sortSquares(int arr[], int n)
{
    // First convert each array elements
    // into its square
    for (int i = 0; i < n; i++)
```

```
        arr[i] = arr[i] * arr[i];

    // Sort an array using "sort STL function "
    sort(arr, arr + n);
}

// Driver program to test above function
int main()
{
    int arr[] = { -6, -3, -1, 2, 4, 5 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Before sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    sortSquares(arr, n);

    cout << "\nAfter Sort " << endl;
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

4.

```
class Solution {
public:
    vector<vector<int>> findDifference(vector<int>& nums1, vector<int>& nums2)
{
        unordered_set<int>set1(nums1.begin(),nums1.end());
        unordered_set<int>set2(nums2.begin(),nums2.end());
        vector<int>distinct_nums1,distinct_nums2;
        for(int num: set1){
            if(set2.count(num)==0){
                distinct_nums1.push_back(num);
            }
        }
        for(int num:set2){
            if(set1.count(num)==0){
                distinct_nums2.push_back(num);
            }
        }
        return {distinct_nums1, distinct_nums2};
```

```
    }
};
```

5.

```
class Solution {
public:
    int findTheDistanceValue(vector<int>& A, vector<int>& B, int d) {
        int ans = 0;
        for (int i = 0; i < A.size(); ++i) {
            bool found = false;
            for (int j = 0; j < B.size() && !found; ++j) {
                if (abs(A[i] - B[j]) <= d) found = true;
            }
            if (!found) ++ans;
        }
        return ans;
    }
};
```

6.

```
#include<bits/stdc++.h>
class Solution {
public:
    vector<int> findDuplicates(vector<int>& nums) {
        vector<int>ans;
        for(int i=0;i<nums.size();++i){
            int index=abs(nums[i])-1;
            if(nums[index]<0)
            ans.push_back(abs(nums[i]));
            nums[index]= -nums[index];
        }
        return ans;
    }
};
```

7.

```
class Solution {
public:
    int findMin(vector<int>& nums) {
```

```
        int n = nums.size();
        if (nums[0] <= nums[n - 1]) return nums[0];
        int left = 0, right = n - 1;
        while (left < right) {
            int mid = (left + right) >> 1;
            if (nums[0] <= nums[mid])
                left = mid + 1;
            else
                right = mid;
        }
        return nums[left];
    }
};
```

8.

```
class Solution {
 public:
  vector<int> findOriginalArray(vector<int>& changed) {
    vector<int> ans;
    queue<int> q;

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

    for (const int num : changed)
      if (!q.empty() && num == q.front()) {
        q.pop();
      } else {
        q.push(num * 2);
        ans.push_back(num);
      }

    return q.empty() ? ans : vector<int>();
  }
};
```