Assignment 04:

1. 867. Transpose Matrix

Given a 2D integer array matrix, return the transpose of matrix.

The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

```
//timecompl O(m*n)

//space O(m*n)

class Solution {

public:

    vector<vector<int>> transpose(vector<vector<int>>& matrix) {

       int r = matrix.size();               // Number of rows

       int c = matrix[0].size();

       vector<vector<int>>ans(c,vector<int>(r));

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

          {

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

              {

                   ans[j][i] = matrix[i][j];



              }

          }

          return ans;

    }

};
```

2. 561. Array Partition

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum.

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

//Space Complexity:O(1)

class Solution {

public:

    int arrayPairSum(vector<int>& nums) {

        //[1,4,3,2]

        //1. Sort [1,2,3,4]

        //2. Optimized Pair (1,2)+(3,4)

        //3. Sum min values: 1+3=4

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

        int sum=0;

        for(int i=0;i<nums.size();i+=2)

        {

            sum=sum+min(nums[i],nums[i

            +1]);

        }

        return sum;

    }

};
```

3. 977. Squares of a Sorted Array

Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

```
class Solution {

public:

    vector<int> sortedSquares(vector<int>& nums) {

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

        {

            nums[i]=nums[i]*nums[i];

        }

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

        return nums;



    }

};
```

4. Find the Difference of Two Arrays

Given two 0-indexed integer arrays nums1 and nums2, return a list answer of size 2 where:

- answer[0] is a list of all distinct integers in nums1 which are not present in nums2.
- answer[1] is a list of all distinct integers in nums2 which are not present in nums1.

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

//Space Complexity: O(n+m)

class Solution {

public:

    vector<vector<int>> findDifference(vector<int>& nums1, vector<int>&
nums2) {

        set<int>set1(nums1.begin(),nums1.end());

        set<int>set2(nums2.begin(),nums2.end());

        vector<vector<int>>res(2);

        for(auto it: set1)

        {

            if(set2.count(it)==0)

            {

                res[0].push_back(it);

            }

        }

        for(auto it: set2)

        {

            if(set1.count(it)==0)

            {
```

```
                res[1].push_back(it);
            }
        }

        return res;
    }
};
```

5. 598. Range Addition II

You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return the number of maximum integers in the matrix after performing all the operations.

```
//Time Complexity: O(n) where n is ops

//Space : O(1)

class Solution {

public:

    int maxCount(int m, int n, vector<vector<int>>& ops) {

        int r=m;

        int c=n;

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

        {

            r=min(r,ops[i][0]);

            c=min(c,ops[i][1]);

        }

        return r*c;



    }

};
```

6. Shuffle the Array

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].

Return the array in the form [x1,y1,x2,y2,...,xn,yn].

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

//Space Complexity: O(n)

class Solution {

public:

    vector<int> shuffle(vector<int>& nums, int n) {

        vector<int>ans;

        int j=n;

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

        {

            ans.push_back(nums[i]);

            ans.push_back(nums[j]);

            j++;

        }

       return ans;

    }

};
```

7. 441. Arranging Coins

You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase may be incomplete.

Given the integer n, return the number of complete rows of the staircase you will build.

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

//Space Complexity:O(1)

class Solution {

public:

    int arrangeCoins(int n) {

        int left=0,right=n;

        while(left<=right)

        {

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

            long k=mid*(mid+1)/2;

            if(k==n)

            {

                return mid;

            }

            else if(k<n)

            {

                left=mid+1;

            }

            else
```

```
            {
                right=mid-1;
            }
        }
        return right;
    }
};
```