(i)1662

class Solution {

    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

         String x = "";

        String y = "";

        for(int i=0; i<word1.length;i++){

            x+=word1[i];

        }

        for(int i=0; i<word2.length;i++){

            y+=word2[i];

        }

        if(x.equals(y)){

            return true;

        }

        return false;

    }

}

(ii)1295

class Solution {

    public int findNumbers(int[] nums) {

          int count=0;

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

            if(Integer.toString(nums[i]).length()%2==0)

                count++;

        }

        return count;

    }

}

(iii)485

class Solution {

    public int findMaxConsecutiveOnes(int[] nums) {

         int l = 0, r = 0, maxOnes = 0;

        while (r < nums.length) {

            while (r < nums.length  && nums[r] == 1) {

                ++r;

            }

            maxOnes = Math.max(maxOnes, r - l);

            ++r;

            l = r;

        }

        return maxOnes;

    }

}

(iv)1920

class Solution {

    public int[] buildArray(int[] nums) {

         int len = nums.length;

        int[] ans = new int[len];

        for(int i = 0; i < len; i++){

            ans[i] = nums[nums[i]];

        }

        return ans;

    }

}

(v)2652

class Solution {

    public int sumOfMultiples(int n) {

        int sum=0;

        for(int i=1;i<=n;i++){

            if(i%3==0 || i%5==0 || i%7==0){

                sum+=i;

            }

        }

        return sum;

    }

}

(vi)1539

class Solution {

    public int findKthPositive(int[] arr, int k) {

     for(int num : arr)

        {

            if(num<=k)

                k++;

            else

                break;

        }

        return k;

    }

}

(vii)53

class Solution {

    public int maxSubArray(int[] nums) {

         int n = nums.length;

        int max = Integer.MIN\_VALUE, sum = 0;

        for(int i=0;i<n;i++){

            sum += nums[i];

            max = Math.max(sum,max);

            if(sum<0) sum = 0;

        }

        return max;

    }

}

(viii)2221

class Solution {

    public int triangularSum(int[] nums) {

          int n = nums.length;

        while(n-- > 1)

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

                nums[i] = (nums[i] + nums[i + 1]) % 10;

        return nums[0];

    }

}

(ix) 121

class Solution {

    public int maxProfit(int[] prices) {

        int buy=Integer.MAX\_VALUE,sell=0;

        for(int i=0;i<prices.length;i++){

            buy=Math.min(buy,prices[i]);

            sell=Math.max(sell,prices[i]-buy);

        }

       return sell;

    }

}

(x) 118

class Solution {

    public List<List<Integer>> generate(int numRows) {

        List<List<Integer>> ans = new ArrayList<>();

    for (int i = 0; i < numRows; ++i) {

      Integer[] temp = new Integer[i + 1];

      Arrays.fill(temp, 1);

      ans.add(Arrays.asList(temp));

    }

    for (int i = 2; i < numRows; ++i)

      for (int j = 1; j < ans.get(i).size() - 1; ++j)

        ans.get(i).set(j, ans.get(i - 1).get(j - 1) + ans.get(i - 1).get(j));

    return ans;

    }

}