LinkedIn - https://www.linkedin.com/in/sakshamarora9575/
Resume- Saksham Arora.pdf
1:1 Connect - https://topmate.io/sakshamarora
Long Term Membership - https://preplaced.in/profile/saksham-arora
Connect with me -
https://whatsapp.com/channel/0029ValNBE0L7UVRRkHU1K2U

Email to me - saksham20189575@gmail.com

Questions - Questions DSA and CP

Resume Template - FAANGPath Simple Template - Overleaf, Online LaTeX Editor

DBMS Notes -
DBMS - August
OS Notes -
OS Notes (August)
SQL Notes -
SQL - August
Computer Network Notes -
Computer Networks - August

27th July (Factorial of the Number)

```
// Import scanner class for taking the input
import java.util.Scanner;

class Solution {

    public static void main(String args[]) {

        // ClassName object = new ClassName();
        Scanner in = new Scanner(System.in);
```

```java
          int n = in.nextInt();

          if(n < 0)
          {
              System.out.println("Error");
          }
          else
          {
              // n! = n * n-1 * n-2 * n-3 ---- 1
              // n! = 1 * 2 * 3 * 4 * 5 .... n

              // 0! = 1

              int factorial = 1;
              for(int i=2; i<=n; i=i+1)
              {
                  factorial = factorial * i;
              }

              System.out.println(factorial);
          }
     }
}
```

27th July (Find the Area of Rectangle)

```java
import java.util.Scanner;

class Solution {

     public static void main(String args[]) {

          Scanner in = new Scanner(System.in);

          int length = in.nextInt();
          int breadth = in.nextInt();

          int area = length * breadth;

          System.out.println(area);
     }
}
```

27th July (Binary To Decimal)

```
import java.util.Scanner;
public class Main {

     public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        int n = in.nextInt();

        int answer = 0;
        int power = 1;

        // 110
        // STEP 1 - lastDigit = 0, power = 1, answer = 0 + 0 * 1
= 0
        // STEP 2 - lastDigit = 1, power = 2, answer = 0 + 1 * 2
= 2
        // STEP 3 - lastDigit = 1, power = 4, answer = 2 + 1 * 4
= 6


        while(n > 0)
        {
            int lastDigit = n%10;

            answer = answer + lastDigit * power;

            n = n/10;
            power = power * 2;
        }

        System.out.println(answer);
    }
}
```

27th July (Sum of Even & Odd)

```java
import java.util.Scanner;
public class Main {

     // 194 / 10 = 19
     public static void main(String[] args) {

          Scanner in = new Scanner(System.in);

          int n = in.nextInt();
          int sumOfEvenDigit = 0;
          int sumOfOddDigit = 0;

          while(n > 0)
          {
              int lastDigit = n%10;

              if(lastDigit % 2 == 0)
              {
                  // Even digit
                  sumOfEvenDigit = sumOfEvenDigit + lastDigit;
              }
              else
              {
                  // odd digit
                  sumOfOddDigit = sumOfOddDigit + lastDigit;
              }

              // Remove the last digit
              n = n / 10;
          }

          System.out.println(sumOfEvenDigit + " " +
sumOfOddDigit);

     }
}
```

27th July (Square Root (Integer))

```
import java.util.Scanner;
public class Main {

     public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        int n = in.nextInt();

        // 1st Approach
        // System.out.println((int)Math.sqrt(n));

        // 2nd Approach
        int answer = 1;

        // n = 10
        // i = 1, 1*1 <= 10, answer = 1
        // i = 2, 2*2 <= 10, answer = 2
        // i = 3, 3*3 <= 10, answer = 3

        // i = 4, 4*4 <= 10(THE LOOP WILL STOP)
        for(int i=1; i*i <= n; i++)
        {
            answer = i;
        }

        System.out.println(answer);

        // Range of square root - 1 ..... n

        // int start = 1;
        // int end = n;
        // int answer = 1;

        // start = 1, end = 10

        // mid = 5, 5 * 5 = 25 <= 10 (No)
        // start = 1, end = 4

        // mid = 2, 2 * 2 <= 10 (Yes)
        // start = 3, end = 4

        // mid = 3, 3 * 3 <= 10 (Yes)
        // start = 4, end = 4

        // mid = 4, 4 * 4 <= 10 (No)
        // start = 4, end = 3 (BINARY SEARCH STOP)

        // while(start <= end)
        // {
        //     int mid = (start + end)/2;

        //     if((long)mid * (long)mid <= n)
        //     {
        //         answer = mid;
        //         start = mid + 1;
        //     }
        //     else
        //     {
        //         end = mid - 1;
        //     }
        // }

        // System.out.println(answer);
     }
}
```

28th July (Count Characters)

```
import java.util.Scanner;
public class Solution {
     public static void main(String arg[]) {

         Scanner in = new Scanner(System.in);

         // next() --> input of single word
         // nextLine() --> input a single line

         int countLower = 0;
         int countDigit = 0;
         int countSpaces = 0;

          int lines = 0;

         // hasNextLine - return true if a next line is present
or not

         while(in.hasNextLine())
         {
             String s = in.nextLine();

         int n = s.length();

         // ch --> '2', '3', '5'
         for(int i=0; i<n; i++)
         {
             char ch = s.charAt(i);

             if(ch >= 'a' && ch <= 'z')
             {
                 countLower++;
             }
             else if(ch >= '0' && ch <= '9')
             {
                 countDigit++;
             }
             else if(ch == ' ')
             {
                 countSpaces++;
             }
                    else if(ch == '\t')
                    {
                         countSpaces++;
                    }
         }

               lines++;
         }

         countSpaces += (lines - 1);

         System.out.println(countLower + " " + countDigit + " "
+ countSpaces);
     }
}
```

28th July (Check Is Fibonacci Number)

```
public class Solution {
     public static boolean CheckFiboNum(int n) {

          if(n == 0)
          {
               return true;
          }

          int arr[] = new int[n+1];

          int firstNumber = 0;
```

```java
          int secondNumber = 1;

          arr[firstNumber] = 1;
          arr[secondNumber] = 1;

          int thirdNumber = firstNumber + secondNumber;
          while(thirdNumber < n+1)
          {
              arr[thirdNumber] = 1;

              firstNumber = secondNumber;
              secondNumber = thirdNumber;

              thirdNumber = firstNumber + secondNumber;
          }

          if(arr[n] == 1)
          {
              return true;
          }
          else
          {
              return false;
          }
     }
}
```

28th July (Greatest Common Divisor)

```java
import java.util.Scanner;
public class Solution {

     public static void main(String[] args)
     {
         Scanner in = new Scanner(System.in);

```

```
        int t = in.nextInt();

        while(t > 0)
        {
            int x = in.nextInt();
            int y = in.nextInt();

            // min(x,y) --- 1
            int gcd = 1;

            for(int i=Math.min(x,y); i>=1; i--)
            {
                if(x%i == 0 && y%i == 0)
                {
                    gcd = i;
                    break;
                }
            }

            System.out.println(gcd);

            t--;
        }
    }
}
```

28th July (Basic Prime number Approach)

```
import java.util.ArrayList;
public class Solution {

    public static ArrayList<Integer> primeNumbersTillN(int N)
    {
        ArrayList<Integer> arr = new ArrayList<>();

        // Prime number have 2 divisors - 1 and itself (1, N)
```

```
          // A number is not prime if it has any divisor from 2
to N-1
          // All divisors they exist in pairs

          for(int i=2; i<=N; i++)
          {
              boolean isPrime = true;

              for(int j=2; j*j <= i; j++)
              {
                  if(i%j == 0)
                  {
                      isPrime = false;
                      break;
                  }
              }

              if(isPrime == true)
              {
                  arr.add(i);
              }
          }

          return arr;
     }
}
```

3rd August (Star Pattern)

```
import java.util.Scanner;

public class Main
{
    public static void main(String arg[])
    {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();

        int stars = 1;
        int spaces = n - 1;

        for(int i=1; i<=n; i++) // n times
        {
            // Print spaces
            for(int j=1; j<=spaces; j++)
            {
                System.out.print(" ");
            }

            // Print stars
            for(int j=1; j<=stars; j++)
            {
                System.out.print("*");
            }

            spaces = spaces - 1;
            stars = stars + 2;

            System.out.println();
        }


    }
}
```

3rd August (Diamond of stars)

```
import java.util.* ;
import java.io.*;
public class Solution {

          public static void printPattern(int n) {
```

```
// Upper part of diamond
int numRowsUpper = n/2 + 1;

int stars = 1;
int spaces = numRowsUpper - 1;

for(int i=1; i<=numRowsUpper; i++)
{
    // Print spaces
    for(int j=1; j<=spaces; j++)
    {
        System.out.print(" ");
    }

    // Print stars
    for(int j=1; j<=stars; j++)
    {
        System.out.print("*");
    }

    stars = stars + 2;
    spaces = spaces - 1;

    System.out.println();
}

// Lower part of diamond
int numRowsLower = n/2;

spaces = 1;
stars = stars - 4;

for(int i=1; i<=numRowsLower; i++)
{
    // Print spaces
    for(int j=1; j<=spaces; j++)
    {
```

```
                System.out.print(" ");
            }

            // Print stars
            for(int j=1; j<=stars; j++)
            {
                System.out.print("*");
            }

            stars = stars - 2;
            spaces = spaces + 1;

            System.out.println();
        }

    }
}
```

3rd August (Print the pattern)

```
import java.util.* ;
import java.io.*;

public class Solution {
        public static String[] NumberPattern(int n) {

            String answer[] = new String[n];

            int turn = 1;
            int i = 0, j = n - 1;

            int number = 1;

            while(i <= j)
            {
                String currentRow = "";

                for(int k=1; k<=n; k++)
                {
                    currentRow = currentRow + number + " ";
                    number++;
                }

                if(turn % 2 != 0)
                {
                    answer[i] = currentRow;
                    i++;
                }
                else
                {
                    answer[j] = currentRow;
                    j--;
                }

                turn++;
            }

            return answer;
        }

}
```

3rd August (Pascal's Triangle Question)

```
import java.util.*;

public class Solution {
    public static void main(String[] args) {
       // write your code logic !!

        Scanner in = new Scanner(System.in);
        int n = in.nextInt();

        ArrayList<ArrayList<Integer>> result = new ArrayList<>();

        ArrayList<Integer> firstRow = new ArrayList<>();
        firstRow.add(1);

        result.add(firstRow);

        // [1], rowIndex = 0
        // [1 1], middleElement = 0, rowIndex = 1
        // [1 2 1], middleElement = 1, rowIndex = 2
        // [1 3 3 1], middleElement = 2, rowIndex = 3

        int numberOfMiddleElement = 0;
        for(int i=1; i<=(n-1); i++)
        {
            ArrayList<Integer> currentRow = new ArrayList<>();
            // first element - 1
            currentRow.add(1);

            // middleElement
            for(int j=1; j<=numberOfMiddleElement; j++)
            {
                int middleElement = result.get(i-1).get(j) + result.get(i-1).get(j-1);
                currentRow.add(middleElement);
            }

            // last element - 1
            currentRow.add(1);

            result.add(currentRow);

            numberOfMiddleElement++;
        }

        int spaces = n - 1;
        for(int i=0; i<n; i++)
        {
            // print spaces
            for(int j=1; j<=spaces; j++)
            {
                System.out.print(" ");
            }

            // print the elements
            for(int j=0; j<result.get(i).size(); j++)
            {
                System.out.print(result.get(i).get(j) + " ");
            }

            System.out.println(); // change the line as well
            spaces--;
        }
    }
}
```

3rd August (Set Bits)

```
import java.util.* ;
import java.io.*;
public class Solution {
        public static int countSetBits(int n) {
                // Write your code here.

                int answer = 0;

                while(n > 0)
                {
                    if(n%2 != 0)
                    {
                        // In binary representation we will have 1
                        answer++;
                    }

                    n = n/2;
                }

                return answer;
        }
}
```

4th August (First K maximum elements)

```
import java.util.* ;
import java.io.*;
import java.util.ArrayList;

public class Solution {
        public static ArrayList<Integer> firstKMax(ArrayList<Integer> arr, int n, int k) {

      // [1, 1, 4, 5, 4], k = 2

      // 1 --> [0, 1]
// 4 --> [2, 4]
// 5 --> [3]

        TreeMap<Integer, ArrayList<Integer>> m = new TreeMap<>();

        for(int i=0; i<arr.size(); i++)
        {
            int element = arr.get(i);

            if(m.containsKey(element) == false)
            {
                ArrayList<Integer> current = new ArrayList<>();
                current.add(i);
                m.put(element, current); // [1, [0]]
            }
            else
            {
                m.get(element).add(i); // [1, [0, 1]]
            }
        }

        ArrayList<Integer> uniqueElements = new ArrayList<>();
        ArrayList<Integer> answer = new ArrayList<>();

        // uniqueElement [1, 4, 5]

        for(Integer key: m.keySet())
        {
            uniqueElements.add(key);
        }

        int index = uniqueElements.size() - 1;

    while(k > 0)
    {
        int element = uniqueElements.get(index);

        for(int i=0; i<m.get(element).size(); i++)
        {
            answer.add(m.get(element).get(i));
        }

        index--;
        k--;
        }

            Collections.sort(answer);

            return answer;
      }
}
```

4th August (Reverse Array)

```
import java.util.* ;
import java.io.*;
import java.util.ArrayList;

public class Solution
{
    public static void reverseArray(ArrayList<Integer> arr, int m)
    {
        int N = arr.size();
        int start = m + 1;
        int end = N - 1;

        while(start <= end)
        {
            int a = arr.get(start);
            int b = arr.get(end);
            int temp = a;
            arr.set(start, b);
            arr.set(end, temp);

            start++;
            end--;
        }
    }
}
```

4th August (Count Vowel, Consonants, Spaces)

```
import java.util.* ;
import java.io.*;
```

```
public class Solution {

    static int[] countVowelsConsonantsSpaces(String s, int n) {

        int vowel = 0;
        int consonants = 0;
        int spaces = 0;

        String res = s.toLowerCase();

        for(int i=0; i<n; i++)
        {
            char currentChar = res.charAt(i);

            if(currentChar == ' ')
            {
                spaces++;
            }
            else if(currentChar == 'a' || currentChar == 'e' || currentChar == 'i' || currentChar == 'o'
            || currentChar == 'u')
            {
                vowel++;
            }
            else
            {
                consonants++;
            }
        }

        int arr[] = new int[3];
        arr[0] = vowel;
        arr[1] = consonants;
        arr[2] = spaces;

        return arr;
    }
}
```

4th (Find anagrams)

```
import java.util.ArrayList;

public class Solution {
```

```
public static boolean compareFre(int freStr[], int frePtr[])
{
    for(int i=0; i<26; i++)
    {
        if(freStr[i] != frePtr[i])
        {
            return false;
        }
    }

    return true;
}

public static ArrayList<Integer> findAnagramsIndices(String str, int n, String ptr, int m){

    int freStr[] = new int[26];
    int frePtr[] = new int[26];

    for(int i=0; i<m; i++)
    {
        char currentChar = ptr.charAt(i);

        frePtr[(currentChar - 'A')]++;
    }

    int i = 0;
    while(i < m)
    {
        char currentChar = str.charAt(i);

        freStr[(currentChar - 'A')]++;
        i++;
    }

    ArrayList<Integer> answer = new ArrayList<>();

    if(compareFre(frePtr, freStr))
    {
        answer.add(0);
    }

    while(i < n)
    {
            char currentChar = str.charAt(i);
            freStr[(currentChar - 'A')]++;

            char previousChar = str.charAt(i - m);
            freStr[(previousChar - 'A')]--;

            if(compareFre(frePtr, freStr))
            {
                answer.add(i - m + 1);
            }

            i++;
        }

        return answer;
    }
}
```

4th August (Rotate Array)

```
import java.util.ArrayList;

public class Solution {
            public static ArrayList<Integer> rotateArray(ArrayList<Integer> arr, int k) {

                int N = arr.size();

                if(k >= n)
                {
                    k = k%n; // k = 1 and k = 6 array is same
                }

                ArrayList<Integer> res = new ArrayList<>();

                // [ 3 4 5]
                //  0 1 2 3 4

                // k = 2  [3 4 5]
                for(int i=k; i<N; i++) // (k ... n-1)
                {
                    res.add(arr.get(i));
                }
                // k = 2  [3 4 5 1 2]
                for(int i=0; i<k; i++) // (0 ... k)
                {
                    res.add(arr.get(i));
                }

                return res;
        }
}
```

10th August (Linear Search)

```
// 2 13 4 1 3 6 28 3 3
// 0 1  2 3 4 5 6  7 8

public class Solution {

    public static int linearSearch(int arr[], int x) {

                    int answer = -1;

                    int N = arr.length;

                    for(int i=0; i<N; i++)
                    {
                        if(arr[i] == x)
                        {
                            answer = i;
                            break;
                        }
                    }

                    return answer;
    }
}
```

10th August (Bubble Sort)

```
import java.util.* ;
import java.io.*;

public class Solution {
    public static void bubbleSort(int[] arr, int n) {

        // 5 4 3 2 1
        // 4 3 2 1 5
        // 3 2 1 4 5
        // 2 1 3 4 5
        // 1 2 3 4 5

        // outer loop - n - 1
        for(int i=0; i<n-1; i++)
        {
            for(int j=0; j<n-1-i; j++)
            {
                if(arr[j] > arr[j + 1])
                {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

    }

}
```

10th August (Insertion Sort)

```
import java.util.* ;
import java.io.*;

public class Solution {

            public static void insertionSort(int n , int[] arr) {

                // 4 5 3 2 1
                // 0 1 2 3 4

                // currentElement = 3, j = 1
                // arr[j] = 5, arr[j] > currentElement (4 5 5 2 1)
                // j = 0, arr[j] = 4, 4 > 3 (4 4 5 2 1)
                // arr[0] = 3 (3 4 5 2 1)
                for(int i=1; i<n; i++)
                {
                    int currentElement = arr[i];
                    int j = i - 1;

                    while(j>=0 && arr[j] > currentElement)
                    {
                        // shifting of elements
                        arr[j + 1] = arr[j];
                    }

                    arr[j + 1] = currentElement;
                }
            }
}
```

10th August (Selection Sort Question)

```
import java.util.* ;
import java.io.*;
public class Solution {
        public static void selectionSort(int arr[], int n) {

            for(int i=0; i<n-1; i++)
            {
                int smallestElement = arr[i];
                int index = i;

                for(int j=i+1; j<n; j++)
                {
                    if(arr[j] < smallestElement)
                    {
                        smallestElement = arr[j];
                        index = j;
                    }
                }

                // place the smallest element to it's correct position
                int temp = arr[i];
                arr[i] = arr[index];
                arr[index] = temp;
            }
        }
}
```

10th August (Aggressive Cows)

```
import java.util.*;

public class Solution {

    public static boolean isPossible(long mid, int[] stalls, int k)
    {
        // mid = 3
        // 1 4 5 6 7 8 9

        int N = stalls.length;

        int count = 1;
        int previous = stalls[0];

        for(int i=1; i<N; i++)
        {
            if(stalls[i] - previous >= mid)
            {
                count++;
                previous = stalls[i];
            }
        }

        if(count >= k)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public static int aggressiveCows(int []stalls, int k) {

        // values --> index of the stall

        // answer will lie between start and end
        Arrays.sort(stalls);
        int N = stalls.length;

        long start = Integer.MAX_VALUE;
        long end = stalls[N - 1] - stalls[0]; // if K = 2

        for(int i=0; i<N-1; i++)
        {
            long currentDifference = stalls[i+1] - stalls[i];

            if(currentDifference < start)
            {
                start = currentDifference;
            }
        }

        long answer = start;

        while(start <= end)
        {
            long mid = (start + end)/2;
            if(isPossible(mid, stalls, k))
            {
                answer = Math.max(answer, mid);
                start = mid + 1; // right direction
            }
            else
            {
                end = mid - 1; // left direction
            }
        }

        return (int)answer;
    }
}
```

11st August (Row with maximum 1)

```
import java.util.ArrayList;
public class Solution
{
    public static int isOnePresentInAnyRow(ArrayList<ArrayList<Integer>> matrix, int col, int n)
    {
        int result = -1;
        for(int i=0; i<n; i++)
        {
            if(matrix.get(i).get(col) == 1)
            {
                result = i;
                break;
            }
        }

        return result;
    }

    public static int maximumOnesRow(ArrayList<ArrayList<Integer>> matrix, int n, int m)
    {
        int start = 0;
        int end = m - 1;

        int answer = -1;

        while(start <= end)
        {
            int mid = (start + end)/2;
            int index = isOnePresentInAnyRow(matrix, mid, n);

            if(index != -1)
            {
                answer = index;
                end = mid - 1;
            }
            else
            {
                start = mid + 1;
            }
        }

        return answer;
    }
}
```

11th August (Maximum Sum subarray)

```
import java.util.* ;
import java.io.*;

public class Solution {

        public static long maxSubarraySum(int[] arr, int n) {

            long currentSum = arr[0];
            long maximumSum = arr[0];

            for(int i=1; i<n; i++)
            {
                long option1 = arr[i] + currentSum;
                long option2 = arr[i];

                currentSum = Math.max(option1, option2);
                maximumSum = Math.max(maximumSum, currentSum);
            }

            if(maximumSum < 0)
            {
                // empty subarray can be chosen
                maximumSum = 0;
            }

            return maximumSum;
        }
}
```

11th August (Matrix Multiplication)

```
import java.util.* ;
import java.io.*;
import java.util.ArrayList;

public class Solution {
    public static ArrayList<ArrayList<Integer>> multiplyMatrices(ArrayList<ArrayList<Integer>>
mat1,
    ArrayList<ArrayList<Integer>> mat2) {

        // mat1 --> a * b
        // mat2 --> b * c

        int a = mat1.size();
        int b = mat1.get(0).size();
        int c = mat2.get(0).size();

        // [1, 2]    [5, 6]
        // [3, 4]    [7, 8]

        ArrayList<ArrayList<Integer>> result = new ArrayList<>();

        for(int i=0; i<a; i++) // rows of mat1
        {
            ArrayList<Integer> currentRow = new ArrayList<>();
            for(int j=0; j<c; j++) // columns of mat2
            {
                // ith row of mat1, jth col of mat2
                int sum = 0;

                for(int k=0; k<b; k++)
                {
                    sum = sum + mat1.get(i).get(k) * mat2.get(k).get(j);
                }

                currentRow.add(sum);
            }

            result.add(currentRow);
        }

        return result;
    }
}
```

11th August (Best time to buy and sell the stocks)

```
import java.util.* ;
import java.io.*;
import java.util.ArrayList;

public class Solution{
    public static int maximumProfit(ArrayList<Integer> prices)
    {
        int n = prices.size();
        int answer = 0;
        int buy = prices.get(0);

        for(int i=1; i<n; i++)
        {
            if(prices.get(i) > buy)
            {
                int currentProfit = prices.get(i) - buy;
                answer = Math.max(currentProfit, answer);
            }
            else
            {
                buy = prices.get(i);
            }
        }

        return answer;
    }
}
```

24th August (Frog Jumps)

```
import java.util.* ;
import java.io.*;
public class Solution {

    const static int N = 100001;
    static int dp[] = new int[N];
    public static int recursion(int index, int n, int heights[])
    {
        if(index == n - 1)
        {
            return 0;
        }

        if(dp[index] != -1) // already calculated values will not be added
        {
            return dp[index];
        }

        // Recursive case ~ Options
        int option1 = 1000000;
        int option2 = 1000000;

        if(index + 1 < n) // jump of 1 step
        {
            option1 = Math.abs(heights[index+1] - heights[index]) + recursion(index + 1, n, heights);
        }
        if(index + 2 < n) // jump of 2 step
        {
            option2 = Math.abs(heights[index+2] - heights[index]) + recursion(index + 2, n, heights);
        }

        int answer = Math.min(option1, option2);
        dp[index] = answer;
        return answer;
    }

    public static int frogJump(int n, int heights[])
    {
        for(int i=0; i<N; i++)
        {
            dp[i] = -1;
        }
        return recursion(0, n, heights);
    }

}
```

24th August (Nth Fibonacci number)

```
import java.util.Scanner;
public class Solution {

    public static long fibo(int n)
    {
        if(n == 1 || n == 2)
        {
            return 1;
        }

        // recursion
        int nth = fibo(n - 1) + fibo(n - 2);
        return nth;
    }
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);

                int n = in.nextInt();

                long first = 1;
                long second = 1;
            int count = n - 2;
            long third = 0;

            // 1 1 2 3 5

            while(count > 0)
            {
                third = first + second;
                first = second;
                second = third;

                count--;
            }

            if(n == 1 || n == 2)
            {
                System.out.println("1");
            }
            else
            {
                System.out.println(third);
            }
        }

}
```

24th August (Balance Parenthesis)

```
import java.util.* ;
import java.io.*;

public class Solution {

    static List<String> answer;

    public static void recursion(int i, int sum, int N, String currentSeq)
    {
        // base case
        if(i == N)
        {
            if(sum == 0)
            {
                // valid sequence
                answer.add(currentSeq);
            }
            return;
        }

        if(sum < N/2)
        {
            // recursive case
            recursion(i + 1, sum + 1, N, currentSeq + "(");
        }

        if(sum >= 1)
        {
            // atleast one opening bracket is kept
            recursion(i + 1, sum - 1, N, currentSeq + ")");
        }

    }

            public static List<String> balancedParantheses(int n) {

                answer = new ArrayList<>()

                // n = 3, ((()))
                recursion(0, 0, 2*n, "");

                return answer;
            }
}
```

25th Aug (Cycle in the linked list)

```
public class Solution {

  public static boolean detectCycle(Node head) {

      Node slow = head;
      Node fast = head;

      while(fast != null && fast.next != null)
      {
          slow = slow.next;
```

```
        fast = fast.next.next;

        if(slow == fast)
        {
            return true;
        }
    }

    return false;
  }
}
```

25th Aug (Intersection of the linked list)

```
public class Solution {
    public static int findIntersection(Node firstHead, Node secondHead) {

        if(firstHead == null || secondHead == null)
        {
            return -1;
        }

        // STEP 1 - find tail/ last node
        Node temp = firstHead;

        // 1 -> 2 -> 3
        while(temp.next != null)
        {
            temp = temp.next;
        }

        temp.next = secondHead;

        Node slow = firstHead;
        Node fast = firstHead;

        boolean isCycle = false;

        while(fast != null && fast.next != null)
        {
            slow = slow.next;
            fast = fast.next.next;

            if(slow == fast)
            {
                isCycle = true;
                // merge point of slow and fast
                break;
            }
        }

        if(isCycle == false)
        {
            return -1; // no intersection point
        }

        fast = firstHead;
        while(fast != slow)
        {
            fast = fast.next;
            slow = slow.next;
        }

        return slow.data;
    }
}
```

31st Aug (Valid Parenthesis)

```
import java.util.*;
public class Solution {
    public static boolean isValidParenthesis(String s) {

        // [({})]
        int l = s.length();

        Stack<Character> st = new Stack<>();

        // {)
        // ))()()((

        for(int i=0; i<l; i++)
        {
            if(s.charAt(i) == '(' || s.charAt(i) == '[' || s.charAt(i) == '{')
            {
                st.push(s.charAt(i));
            }
            else
            {
                if(st.isEmpty())
                {
                    return false;
                }

                if(s.charAt(i) == ')')
                {
                    if(st.peek() != '(')
                    {
                        return false;
                    }
                }

                if(s.charAt(i) == '}')
                {
                    if(st.peek() != '{')
                    {
                        return false;
                    }
                }

                if(s.charAt(i) == ']')
                {
                    if(st.peek() != '[') // find the topmost element
                    {
                        return false;
                    }
                }

                // pop - remove the topmost element
                st.pop();
            }
        }

        return st.isEmpty();
    }
}
```

31st Aug (Next Smaller Element)

```
import java.util.*;
import java.io.*;

public class Solution{
  static ArrayList<Integer> nextSmallerElement(ArrayList<Integer> arr, int n){

      ArrayList<Integer> answer = new ArrayList<>();

      for(int i=0; i<n; i++)
      {
          answer.add(-1);
      }

      Stack<Integer> st = new Stack<>();

      for(int i=0; i<n; i++)
      {
          while(!st.isEmpty() && arr.get(i) < arr.get(st.peek()))
          {
              answer.set(st.peek(), arr.get(i));
              st.pop();
          }

          st.push(i);
      }

      return answer;
  }
}
```

31st Aug (Delete middle element from stack)

```
import java.util.* ;
import java.io.*;
public class Solution {
  public static void deleteMiddle(Stack<Integer> inputStack, int N) {
      Stack<Integer> st = new Stack<>();

      int m = (N+1)/2 - 1;

      while(m > 0 && !inputStack.isEmpty())
      {
          st.push(inputStack.pop());
          m--;
      }

      if(!inputStack.isEmpty())
      {
          inputStack.pop();
      }

      while(!st.isEmpty())
      {
          inputStack.push(st.pop());
      }
  }
}
```

31st Aug (Reverse first K elements from queue)

```
import java.util.* ;
import java.io.*;
public class Solution
```

```
{
 public static Queue<Integer> reverseElements(Queue<Integer> q, int k)
 {
     Stack<Integer> first = new Stack<>();
     Queue<Integer> answer = new LinkedList<>();

     while(k > 0)
     {
         first.push(q.poll());
         k--;
     }

     while(!first.isEmpty())
     {
         answer.add(first.pop());
     }

     while(!q.isEmpty())
     {
         answer.add(q.poll());
     }

     return answer;
 }
}
```

1st Sept (Buy and sell stock)

```
import java.util.* ;
import java.io.*;
import java.util.ArrayList;

public class Solution{
    public static int maximumProfit(ArrayList<Integer> prices){

        int n = prices.size();
        int answer = 0;
        int minimum = prices.get(0);

        for(int i=1; i<n; i++)
        {
            if(prices.get(i) > minimum)
            {
                answer = Math.max(answer, prices.get(i) - minimum);
            }
            else
            {
                minimum = prices.get(i);
            }
        }

        return answer;
    }
}
```

1st Sept (Sliding window maximum)

```
import java.util.*;
public class Solution {
            public static int[] maxSlidingWindow(int[] arr, int n, int k) {

                int answer[] = new int[n - k + 1];
                Deque<Integer> dq = new LinkedList<>();

                int i = 0;
                while(i < k)
                {
                                        // arr[i] is bigger than previous element
                                        // arr[i] is useless element, we will remove it
                    while(!dq.isEmpty() && arr[i] > arr[dq.peekLast()])
                    {
                        dq.removeLast();
                    }

                    dq.add(i);
                    i++;
                }

                while(i < n)
                {
                    answer[i - k] = arr[dq.peekFirst()];
                    // all the elements out of the window will be removed
                    while(!dq.isEmpty() && dq.peekFirst() <= (i - k))
                    {
                        dq.removeFirst();
                    }

                    // current new element to be added
                                        // arr[i] is bigger than previous element
                                        // arr[i] is useless element, we will remove it
                    while(!dq.isEmpty() && arr[i] > arr[dq.peekLast()])
                    {
                        dq.removeLast();
                    }

                    dq.add(i);

                    i++;
                }

                answer[i - k] = arr[dq.peekFirst()];

                return answer;
            }
}
```

1st Sept (Trapping Rain water)

```
public class Solution {
    public static long getTrappedWater(long []arr, int n) {

        long prefixMax[] = new long[n];
        long suffixMax[] = new long[n];

        long max = arr[0];
        for(int i=0; i<n; i++)
        {
            max = Math.max(arr[i], max);
            prefixMax[i] = max;
        }

        max = arr[n - 1];
        for(int i=n-1; i>=0; i--)
        {
            max = Math.max(arr[i], max);
            suffixMax[i] = max;
        }

        long answer = 0;
        for(int i=1; i<=n-2; i++)
        {
            long minHeight = (Math.min(prefixMax[i-1], suffixMax[i+1]));
            if(minHeight > arr[i])
            {
                answer +=  (minHeight - arr[i]);
            }
        }

        return answer;
    }
}
```

1st Sept (Anagram Question)

```
public class Solution {
    public static boolean isAnagram(String str1, String str2) {

        int fre[] = new int[26];

        for(int i=0; i<str1.length(); i++)
        {
            int ascii = str1.charAt(i) - 'a';
            fre[ascii]++;
        }

        for(int i=0; i<str2.length(); i++)
        {
            int ascii = str2.charAt(i) - 'a';
            fre[ascii]--;
        }

        for(int i=0; i<26; i++)
        {
            if(fre[i] != 0)
            {
                return false;
            }
        }

        return true;
    }

}
```

1st Sept (Celebrity Problem)

```
import java.util.* ;
import java.io.*;

public class Solution {
   public static int findCelebrity(int n) {

       Stack<Integer> st = new Stack<>();

       for(int i=0; i<=n-1; i++)
       {
           st.push(i);
       }

       // 3 2 1

       while(st.size() >= 2)
       {
           int first = st.pop();
           int second = st.pop();

           if(Runner.knows(first, second) == true)
           {
               // first cannot be the Celebrity
               if(Runner.knows(second, first) != true)
               {
                   st.push(second);
               }
           }
```

```
            else
            {
                // second cannot be Celebrity
                if(Runner.knows(second, first) == true)
                {
                    st.push(first);
                }
            }
        }

        if(st.isEmpty())
        {
            return -1;
        }
        else
        {
            int isCelebrity = 1;
            for(int i=0; i<n; i++)
            {
                if(i!=st.peek() && Runner.knows(i, st.peek()) != true)
                {
                    isCelebrity = 0;
                }
            }

            if(isCelebrity == 0)
            {
                return -1;
            }
            else
            {
                return st.peek();
            }
        }

    }
```

```
}
```