3rd October (Big Light Question)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here

            Scanner in = new Scanner(System.in);

            int height1 = in.nextInt();
            int height2 = in.nextInt();

            int speed1 = in.nextInt();
            int speed2 = in.nextInt();

            if(speed1 == speed2)
            {
                System.out.println("false");
            }
            else // speed2!=speed1
            {
                int numerator = (height2 - height1);
                int denominator = (speed1 - speed2);

                if(numerator%denominator == 0)
                {
                    System.out.println("true");
                }
                else
                {
                    System.out.println("false");
                }
            }
      }
}
```

3rd October (Which Angled Triangle Question)

```java
// A triangle is acute-angled, if twice the square of the largest side is
less than the sum of squares of all the sides.

// A triangle is obtuse-angled, if twice the square of its largest side is
greater than the sum of squares of all the sides.

// A triangle is right-angled, if twice the square of its largest side is
exactly equal to the sum of squares of all the sides.
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here

            Scanner in = new Scanner(System.in);

            int first = in.nextInt();
            int second = in.nextInt();
            int third = in.nextInt(); // largest one

        // sum of squares of all the sides
            int sumOfSquares = first*first + second*second + third*third;


            // 2*MAX*MAX  --> lhs
            // max*max + a*a + b*b --> rhs

            // max*max --> lhs
            // a*a + b*b --> rhs

            if(2*third*third < sumOfSquares)
            {
                System.out.println("1"); // acute-angled
            }
            else if(2*third*third == sumOfSquares)
            {
                System.out.println("2"); // right-angled
            }
            else
            {
                System.out.println("3");
            }
      }
}
```

3rd October (Type Casting)

```
import java.util.*;
public class Main
{
      public static void main(String[] args)
      {
            Scanner anything = new Scanner(System.in);

            long sum = 500000000000l;
            long val = 1000000000000001l; // l -> long
            System.out.println(sum);
            System.out.println(val);

            int a = 5;
            double b = a; // implicit typecasting (compiler)

            double z = a + b;

            double c = 5.9;
            int d = (int)c; // explicit typecasting (user forcily)

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

3rd October (Fahrenheit to Celsius Question)

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

public class Main
{
```

```java
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);
        int tempInFah = in.nextInt();
        int tempInCel = ((tempInFah - 32)*5)/9;
        System.out.println(tempInCel);
      }
}
```

3rd October (Celsius to Fahrenheit)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);
        int tempInCel = in.nextInt();
        int tempInFah = (tempInCel*9)/5 + 32;
        System.out.println(tempInFah);
      }
}
```

4th October (Leap Year)

```java
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int year = in.nextInt();
        // A Leap year is the year that is multiple of 4 except multiples
of 100 which are not multiples of 400 are not leap years.

            // If year is not multiple of 4 it not be a leap year
            // If year is a multiple of 4
            // --> IF year is multiple of 100,
            //
            // --> If year is not multiple of 100, then definately a leap
year

             if(year%4==0)
             {
                 if(year%100 == 0)
                 {
                     if(year%400 == 0)
                     {
                         System.out.println(1); // 2000
                     }
                     else
                     {
                         System.out.println(0); // 1900,1700
                     }
                 }
                 else
                 {
                     System.out.println(1); // 2016,2012,2020
                 }
             }
             else
             {
                 System.out.println(0); // 1111
             }

      }
}
```

4th October(Number of Days)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
```

```java
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int month = in.nextInt();

            if(month == 2)
            {
                System.out.println("28");
            }
            else if(month == 1 || month == 3 || month == 5 || month == 7 ||
month == 8 || month == 10 || month == 12)
            {
                System.out.println("31");
            }
            else
            {
                System.out.println("30");
            }

      }
}
```

4th October (Loops Discussion)

```java
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

//          for(initialization; testCondition; updateExpression)
//          {
//              anything
//          }

        // int k = 5;
        // k = k++;

        // System.out.println(k);

            // testCondition -> how many times loop will run

            //
            // Infinite loop
//          for(int k=0;k<5;k--)
//          {
//              System.out.println(k);
//          }

//          for(int i=0;i<5;i++)
//          {
//
//              System.out.println(i);
//            //  continue;
//          }


            // Operations = 25
            for(int i=0;i<5;i++) // LOOP 1 - 5 times
            {
                for(int j=0;j<5;j++) // LOOP 2 - It runs once then it
breaks
                {
                    System.out.print(j+" "); // anything
                    //  break; // it will break the loop
                  //  continue; // all lines written after the continue
statement they will get skipped
                }
                continue;
                //  break;
            }
      }
}
```

4th October (Star Pattern)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();

            // Row = 0, Loop 2 will run 1 time
          // Row = 1, Loop 2 will run 2 time
          // Row = 2, Loop 2 will run 3 time
          // Row = 3, Loop 2 will run 4 time
          // Row = (n-1), Loop 2 will run n time
            for(int row=1;row<=n;row++)
            {
                for(int col=1;col<=row;col++) // Loop 2
                {
                    System.out.print("*");
                }
                System.out.println();
            }
      }
}
```

4th October (HCF)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int a = in.nextInt();
            int b = in.nextInt();

            int minimumOfTwo = Math.min(a,b); // (a<b) ? a:b

            int hcf = 1;
            // Operations = Minimum of a and b
            for(int i=2;i<=minimumOfTwo;i++)
            {
                if(a%i==0 && b%i==0) // whether i divides both a and b
                {
                    hcf = i; // largest value of i obeying the if condition
that will be stored
                }
            }

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

4th October (Minimum & Maximum of Two Numbers)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int a = in.nextInt();
            int b = in.nextInt();
            int c = in.nextInt();

            // Math.min(a,b,c) --> wrong
//          int d = (a>b && a>c) ? a : ((b>c && b>a) ? b: c);

        int d = (int)(Math.max(a,Math.max(b,c)));

      }
}
```

4th October (Peak Element)

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

```java
public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();
            int a[] = new int[n];

            for(int i=0;i<n;i++)
            {
                a[i] = in.nextInt();
            }

            int peakElementIndex = -1;

            // STEP 1
            if(n>1 && a[0]>a[1]) // left boundary
          {
              peakElementIndex = 0; // 0 index
          }
          else
          {
              // STEP 2
              for(int index=1;index<=(n-2);index++)
            {
                if(a[index]>a[index-1] && a[index]>a[index+1])
                {
                    peakElementIndex = index;
                    break; // first peak element
                }
            }

                // STEP 3
                // If peak element is already found it will not be -1
          if(peakElementIndex == -1 && n>=2 && a[n-1]>a[n-2]) // right
boundary
          {
              peakElementIndex = n-1; // last index
          }
          }
```

```
            if(n==1)
              {
                  peakElementIndex = 0;
              }

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

4th October (Armstrong Number)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);


            for(int i=1;i<=500;i++)
            {
                int sum = 0;
                int n = i; // if I change
            while(n>0)
            {
                int lastDigit = n%10; // finding the last digit
                sum = sum + lastDigit*lastDigit*lastDigit; // adding cube
of this digit
                n = n/10; // removing the last digit
            }

            // sum will be having sum of all digits
            if(sum == i) // if sum is as same as original number then it
will be amstrong number
            {
                System.out.println(i);
            }
            }
      }
}
```

6th October (Use of arrays)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int a[] = new int[n];

        int maximumOfNumbers = 0;
        for(int i=0;i<n;i++)
        {
            a[i] = in.nextInt(); // we will input the number given by the
user
            if(a[i] > maximumOfNumbers)
            {
                maximumOfNumbers = a[i];
            }
            // maximumOfNumbers = Math.max(maximumOfNumbers, x);
        }

        System.out.println(maximumOfNumbers);

        // whenever we require of group of numbers in future and they
belong to similiar datatype
        for(int i=0;i<n;i++)
        {
            System.out.print(a[i]+" ");
        }
    }
}
```

6th October(hasNextInt())

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        // If there is integer then it will return true
        // If there is not integer then it will return false
        boolean ans = in.hasNextInt();
        System.out.println(ans);
    }
}
```

6th October (Maximum of number when court of numbers is not known)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        // Limit of arrays --> size of arrays is fixed

        int maximumOfNumber = Integer.MIN_VALUE; // -10^18, 10^18
        // Integer.MAX_VALUE
        while(in.hasNextInt())
        {
            int x = in.nextInt();
            maximumOfNumber = Math.max(maximumOfNumber, x);
            // System.out.print(x+" ");
        }

        System.out.println(maximumOfNumber);
        // ArrayList --> If size is not known before then we will use
ArrayList
    }
}
```

6th October (Memory management of array)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int x = 10;
        int y = x;
        x = 15;
        // System.out.println(x+" "+y);

        int a[] = new int[3];
        a[0] = -1;

        a[0] = -a[0];
        a[1] = 2;
        a[2] = 3;

        // int temp[] = a;
        int temp[] = a;
        temp[2] = 30;

        System.out.println(temp[2]);
        System.out.println(a[2]);

        // a[1]

        // 1. Instance variable is created in stack memory
        // 2. Actual values are stored in heap memory
    }
}
```

6th October (Sum of Numbers question)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        long n = in.nextLong();

        long sum = n*(n+1l)/2l;
        System.out.println(sum);
    }
}
```

6th October (Largest Number At Least Twice of Others)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int a[] = new int[n];

        int indexOfMaximumNumber = 0;
        int maximumNumber = Integer.MIN_VALUE; // ~ -2*10^9
        for(int i=0;i<n;i++)
        {
            a[i] = in.nextInt(); // we will input the number given by the
user
            if(a[i] > maximumNumber)
            {
                maximumNumber = a[i];
                indexOfMaximumNumber = i;
            }
        }

        // assumption --> currently the maximum number is >= 2 * all number
expect maximum number
        boolean flag = true;

        for(int i=0;i<n;i++)
        {
            if(a[i]!= maximumNumber) //
            {
                if(2*a[i]>maximumNumber) // condition for breaking the
assumption
                {
                    flag = false;
                    break;
                }
            }
        }

        if(flag == true)
        {
            System.out.println(indexOfMaximumNumber);
        }
        else
        {
            System.out.println(-1);
        }
    }
}
```

6th October (Power of a Number)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner sc=new Scanner(System.in);
                int a = sc.nextInt();
                int b = sc.nextInt();
                int power = 1;

                for (int i=1; i<=b; i++) { //
                        power = power*a;
                }
                System.out.println(power);
      }
}
```

6th October (2nd Largest Number)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int a[] = new int[n];

        for(int i=0;i<n;i++)
        {
            a[i] = in.nextInt(); // we will input the number given by the
user
        }

        int largest = Integer.MIN_VALUE;
        int secondLargest = Integer.MIN_VALUE;
        for(int i=0;i<n;i++)
        {
            if(a[i]>=largest)
            {
                secondLargest = largest;
                largest = a[i];
            }
            else if(a[i]<largest && a[i]>secondLargest)
            {
                secondLargest = a[i];
            }
        }

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

6th October (Sorted Insert Position)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                int arr[]= new int[n];
                for (int i=0;i<n;i++)
                        {
                                arr[i]= sc.nextInt();
                        }
                int B = sc.nextInt();
                int index=n;
                for (int i=0;i<n;i++)
                        {
                                if (arr[i]>= B)
                                {
                                        index=i;
                                        break;
                                }
                        }
                System.out.println(index);
        }
}
```

7th Oct (Factorial Question)

```
import java.util.*;

public class Main
{
        public static void main (String[] args) throws java.lang.Exception
        {
                Scanner sc = new Scanner(System.in);
                long n = sc.nextInt();
                long ans = 1l;

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

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

7th Oct (2D arrays Basic)

```
import java.util.*;

public class Main
{
        public static void main (String[] args) throws java.lang.Exception
        {
                Scanner in = new Scanner(System.in);

                int row = in.nextInt();
                int col = in.nextInt();
                // datatype arrayName[][] = new
```

```
datatype[numberOfRows][numberOfColumns]
        int a[][] = new int[row][col];

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                a[i][j] = in.nextInt();
            }
        }

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                System.out.print(a[i][j]+" ");
            }
            System.out.println();
        }
        System.out.println();

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                System.out.print(a[j][i]+" ");
            }
            System.out.println();
        }

        System.out.println();
        int sum = 0;
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                if(i==j || (i+j) == (row-1))
                {
                    sum+=a[i][j];
                    System.out.print(a[i][j]);
                }
                else
                {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }

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

7th Oct (Alternate Manner Matrix Traversal)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner in = new Scanner(System.in);

            int row = in.nextInt();
            int col = in.nextInt();
            // datatype arrayName[][] = new
datatype[numberOfRows][numberOfColumns]
        int a[][] = new int[row][col];

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                a[i][j] = in.nextInt();
            }
        }

        for(int i=0;i<row;i++)
        {
            if(i%2==0)
            {
                // left to right
                for(int j=0;j<col;j++)
                {
                    System.out.print(a[i][j]+" ");
                }
            }
            else
            {
                // right to left
                for(int j=(col-1);j>=0;j--)
                {
                    System.out.print(a[i][j]+" ");
                }
            }
            // System.out.println();
        }
      }
}
```

7th Oct (Boolean Matrix Problem)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner in = new Scanner(System.in);

            int row = in.nextInt();
            int col = in.nextInt();
            // datatype arrayName[][] = new
datatype[numberOfRows][numberOfColumns]
        int a[][] = new int[row][col];

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                a[i][j] = in.nextInt();
            }
        }

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                if(a[i][j] == 1)
                {
                    for(int k=0;k<col;k++) // we are iterating again from
the start
                    {
                        a[i][k] = 1;
                    }
                    break;
                }
            }
        }

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                System.out.print(a[i][j]+" ");
            }
            System.out.println();
        }

      }
}
```

7th October (Reverse the array)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();

            int a[] = new int[n];

            for(int i=0;i<n;i++)
            {
                a[i] = in.nextInt();
            }

            int start = 0;
            int end = n-1;

            while(start < end)
            {
                // reverse the array
                // interchange a[start] and a[end]
                int temp = a[end];
                a[end] = a[start];
                a[start] = temp;

                start++;
                end--;
            }

            for(int i=0;i<n;i++)
            {
                System.out.print(a[i]+" ");
            }
      }
}
```

7th October (Rotate the matrix by 90 degree)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner in = new Scanner(System.in);

            int row = in.nextInt();
            int col = in.nextInt();
            // datatype arrayName[][] = new
datatype[numberOfRows][numberOfColumns]
        int a[][] = new int[row][col];

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                a[i][j] = in.nextInt();
            }
        }

        // transpose of the matrix
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<i;j++)
            {
                int temp = a[j][i];
                a[j][i] = a[i][j];
                a[i][j] = temp;
            }
        }


        for(int i=0;i<row;i++)
        {
            // for every row do the reverse operation
            int start = 0;
            int end = col-1;

            while(start < end)
            {
                // reverse the array
                // interchange a[start] and a[end]
                int temp = a[i][end];
                a[i][end] = a[i][start];
                a[i][start] = temp;

                start++;
                end--;
            }

        }

        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                System.out.print(a[i][j]+" ");
            }
            System.out.println();
        }


      }
}
```

7th October (Marc Cakewalk Question)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();

            long a[] = new long[n];

            for(int i=0;i<n;i++)
            {
                a[i] = in.nextLong();
            }

            // a^b --> (int)(Math.pow(a,b))

            long sum = 0;
            Arrays.sort(a); // arrange in asc order

        // long pow = 1;
            int j = 0;
            for(int i=(n-1);i>=0;i--)
            {
                //  sum = (sum + a[i]*pow);
                sum = (sum + a[i]*(long)(Math.pow(2,j)));
                j++;
                //  pow = pow*2l;
            }
            System.out.println(sum);
      }
}
```

9th October (Mirror Right angle Triangle)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);
            int t = in.nextInt();

            while(t>0)
            {
                int n = in.nextInt(); // we will input n for every t

                // logic
                // 1. n number of rows
                for(int row=1;row<=n;row++)
                {
                    // how many spaces I need to print for ith row (n-row)
                    for(int j=0;j<(n-row);j++)
                    {
                        System.out.print(" ");
                    }

                    // how many values we need to print for ith row (row)
                    for(int j=0;j<row;j++)
                    {
                        System.out.print(n-row+1); // formula for value
been printed -> n-row+1
                    }

                    System.out.println();
                }
                t--; // decreasing the test case so that loop starts
            }
      }
}
```

9th October (House Construction 2)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        // f = c+d
        //cost= c+ d+i
        Scanner scn = new Scanner(System.in);

        int house= scn.nextInt();

        double construct = 0.6*house;
        double des = 0.2*house;
        int fndtn= (int)des + (int)construct;

        System.out.println(fndtn);

        if(fndtn <= 100)
        {
            System.out.println("D");
        }
        else if(fndtn <=1000)
        {
            System.out.println("C");

        }
        else if(fndtn <= 10000)
        {
            System.out.println("B");
```

```java
         }else if(fndtn <= 100000)
         {
             System.out.println("A");

         }else if(100000 < fndtn)
         {
             System.out.println("S");
         }
       }
}
```

9th October (LCM Contest)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner input=new Scanner(System.in);
        long a = input.nextLong();
        long b = input.nextLong();

            long minimumOfTwo = Math.min(a,b); // (a<b) ? a:b

            long hcf = 1;
            // Operations = Minimum of a and b
            for(int i=2;i<=(minimumOfTwo);i++)
            {
                if(a%i==0 && b%i==0) // whether i divides both a and b
                {
                    hcf = i; // largest value of i obeying the if condition
that will be stored
                }
            }

            long lcm = (a*b)/hcf;
            System.out.println(lcm);

      }
}
```

9th October (Subarray sum divisible by K)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int k = in.nextInt();
        int a[] = new int[n];

        for(int i=0;i<n;i++)
        {
            a[i] = in.nextInt();
        }

        int countOfSubarrays = 0;
        for(int start=0;start<n;start++)
        {
            // ending point = start - n-1
            int sum = 0;
            for(int end=start;end<=(n-1);end++) // when this loop is moving
starting point is fixed
            {
                sum = sum + a[end]; // sum from start to end
                if(sum % k == 0)
                {
                    countOfSubarrays++;
                }
            }
        }

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

9th October (Divisible Sum Pairs)

```java
import java.io.*;
import java.util.*;
public class Main
{
    public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int k = sc.nextInt();

    int arr[] = new int[n];
    for(int i=0;i<n;i++){
      arr[i] = sc.nextInt();
    }
       int ans =0;
       for(int i=0;i<n;i++){
         for(int j=i+1;j<n;j++){
           if((arr[i]+arr[j])%k==0){
             ans++;
           }
         }
       }
    System.out.println(ans);
  }
}
```

9th October (Print All Subarrays)

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int a[] = new int[n];

        for(int i=0;i<n;i++)
        {
            a[i] = in.nextInt();
        }

        for(int start=0;start<n;start++)
        {
            // ending point = start - n-1
            for(int end=start;end<=(n-1);end++) // when this loop is moving
starting point is fixed
            {
                for(int i=start; i<=end; i++)
                {
                    System.out.print(a[i]+" ");
                }
                System.out.println();
            }
        }
    }
}
```

10th Oct (Strings - 1)

```
public class Main
{
      public static void main(String[] args) {

//          int arr[] = {1,5,3,4,5,6,7,8,9,5};

//         // convert char array to string
//         char crr[] = {'1','2','3'};
//         String s = new String(crr); // s = "123"

//         // create any string
//         String b = "sachin";

//         String s = new String("Virat");

        // String s = "Sachin";
        // s = s.concat("100"); // you need to enter strings only

        String s = "Sachin";
        s = 100 + s + 'A' + 100+100; // Both characters and integers they
are converted to string

        int b = 100+100;

        System.out.println(b);
        System.out.println(s);
      }
}
```

10th Oct (Strings - 2)

```
public class Main
{
      public static void main(String[] args) {

//           int arr[] = {1,5,3,4,5,6,7,8,9,5};

//          // convert char array to string
//          char crr[] = {'1','2','3'};
//          String s = new String(crr); // s = "123"

//          // create any string
//          String b = "sachin";

//          String s = new String("Virat");

        // String s = "Sachin";
        // s = s.concat("100"); // you need to enter strings only

        // String s = "Sachin";
        // s = 5.0 + 100 + 'a' + s + 100 + 5.0 + 'a'; // Both characters
and integers they are converted to string

        // int b = 100+100;

        // System.out.println(b);
        // System.out.println(s);

        // 'a' - 'z' --> 97 - 122
        // 'A' - 'Z' --> 65 - 90
        // '0' - '9' --> 48 - 57
        // ' ' (space) --> 32

        char e = 57;
        System.out.println(e);

        // char to integer
        char x = '_';
        int y = x;
        System.out.println(y);

        // integer to character
        int z = 65;
        char a = 1+'A';
        System.out.println(a);

      }
}
```

10th Oct (Strings - 3)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner sc=new Scanner(System.in);

        int arr[] = new int[10];
        int sizeOfArray = arr.length; // without bracket
        // arr[i] --> ith element of arrays

        String s = sc.next(); // for taking input a single word
        // String a = sc.nextLine(); //  for taking input sentence

        int size = s.length(); // with bracket

        for(int i=0;i<s.length();i++)
        {
            // s.charAt(i) --> ith character
            // ith character present in the string
            System.out.println(s.charAt(i)); // s[i]
        }


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

10th Oct (Strings - 4)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner sc=new Scanner(System.in);

        // char b = sc.next().charAt(0); // using this line you can input a
single character

        int l = sc.nextInt();

        String s = sc.next(); // input a word written by word

        // String to a character array
        char arr[] = new char[l];
        for(int i=0;i<l;i++)
        {
            arr[i] = s.charAt(i); // s.charAt(i) will return the ith
character
        }

        char brr[] = s.toCharArray(); // inbuilt function

        double a[] = new double[3];
        a[0] = 1.0;
        a[1] = 2.0;
        a[2] = 3.0;
        System.out.println(a);

        System.out.println(s);
        System.out.println(arr);
        System.out.println(brr);
      }
}
```

11th Oct (Ptice Question)

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // your code here
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        String answerKey = in.next();

        // ABC ABC ABC
        // 012 345 678
        // 123 456 789

        String pattern1 = "ABC";
        int marks1 = 0;

        for(int i=0;i<n;i++)
        {
            if(answerKey.charAt(i) == pattern1.charAt((i%3)))
            {
                marks1++;
            }
        }

        // BABC BABC
        // 0123 4567

        String pattern2 = "BABC";
        int marks2 = 0;

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

```java
        {
            if(answerKey.charAt(i) == pattern2.charAt((i%4)))
            {
                marks2++;
            }
        }

        String pattern3 = "CCAABB";
        int marks3 = 0;

        for(int i=0;i<n;i++)
        {
            if(answerKey.charAt(i) == pattern3.charAt((i%6)))
            {
                marks3++;
            }
        }

        int maximumMarks = Math.max(marks1, Math.max(marks2,marks3));
        System.out.println(maximumMarks);
        if(maximumMarks == marks1)
        {
            System.out.println("Adrian");
        }
        if(maximumMarks == marks2)
        {
            System.out.println("Bruno");
        }
        if(maximumMarks == marks3)
        {
            System.out.println("Goran");
        }
    }
}
```

11th Oct (Strings - 5)

```java
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // your code here
        Scanner in = new Scanner(System.in);

        String s1 = "abc";
        String s2 = "deefeee"; // z = 122, a = 97

        // Find out first character which is different --> subtract their
ascii value
        // If first min_length characters are same --> subtract the length


        // int ans = s1.compareTo(s2); store it

        // first > second --> +ve
        // first < second --> -ve
        // first == second --> 0

        System.out.println(s1.compareTo(s2));
        System.out.println(s2.compareTo(s1));

        System.out.println(s1.equals(s2));

        s1 = s1.concat(s2);

        System.out.println(s1);

        System.out.println(s1.indexOf("feee")); // substring, character or
that ascii value

        System.out.println(s1.substring(1,5));

    }
}
```

11th Oct (**Shuffle String)**

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // your code here
        Scanner in = new Scanner(System.in);

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String s=sc.next();
        int position[] = new int[n];
        for(int i=0;i<n;i++)
        {
            position[i]=sc.nextInt();
        }

        char[] ans = new char[n];
        for(int i=0;i<n;i++)
        {
            ans[position[i]] = s.charAt(i);
        }
        System.out.print(ans);


    }
}
```

11th Oct (Hey Question)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner sc=new Scanner(System.in);
                String input = sc.next();
                int size = input.length();

                System.out.print(input.charAt(0)); // h

                int numberOfE = size-2;
                int iteration = 2*numberOfE;
                while (iteration > 0) {
                    System.out.print('e');
                    iteration--;
                }

                // ans+="ee";
                // ans = ans+"ee"
                // ans = "ee"+ans
                System.out.print(input.charAt(size-1)); // y
      }
}
```

11th Oct (Time Conversion)

```
import java.util.*;

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            Scanner sc=new Scanner(System.in);
        String time = sc.next();

        int len = time.length();

        int hours = (time.charAt(0)-'0')*10 + (time.charAt(1)-'0');

        if(time.charAt(len-2) == 'A') // AM time
        {
            if(hours == 12)
            {
                System.out.print("00"+time.substring(2,len-2));
            }
            else
            {
                System.out.print(time.substring(0,len-2));
            }
        }
        else // PM time
        {
            if(hours == 12)
            {
                System.out.print(time.substring(0,len-2));
            }
            else
            {
                hours+=12;
                System.out.print(hours+time.substring(2,len-2));
            }
        }
      }
}
```

13th (Functions)

```
import java.util.*;

public class Main
{
    // public static int[] abc()
    // {
    //     int ans[] = new int[10];

    //     // logic

    //     return ans;
    // }



    public static void FirstFunction()
    {
        int a = 10;
        // some part of code (Logic )
        // Logic part
        System.out.println(a+" First");
    }
    public static void SecondFunction()
    {
        int a = 5;
        // some part of code
        FirstFunction();
        System.out.println(a + " Second");
    }
    public static void ThirdFunction()
    {
        // some part of code
        SecondFunction();
        System.out.println("Third");
        FirstFunction();
    }

    // access_specifier static return_type function_name(parameter list
(data_type name))
    // {

    // }

    public static void abc(int a,float b,int c)
    {
        System.out.println(a);
    }

    public static void abc(int a,int b,int c)
    {
        System.out.println(a);
    }

    public static void abc(int a,int b,String c)
    {
        // logic
        a = 5;
        b = 10;

        int e = 10;
        System.out.println(c+", "+a+", "+b);
        return;
    }

    //execution starts from the main function
      public static void main(String[] args) {

         // for(int i=0;i<args.length;i++)
         // {
         //     System.out.println(args[i]);
         // }


//          ThirdFunction();

//          int[] ans = abc(); //
        abc(100,150.0,784);
        abc(150,178,1478);
        String e = "SACHIN";
        abc(100,150,"VIRAT");
//          System.out.println(ans);

            // reusablitily of code
            // clearly understand the code


//          SecondFunction();
//          ThirdFunction();

//          // some code

//          FirstFunction();

//          // some code

//          FirstFunction();
      }
}
```

13th (First and Last position using functions)

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

public class Main
{
    public static void FirstAndLastPosition(int arr[],int n,int target)
    {
        int firstIndex = -1;
        int secondIndex = -1;

        for(int i=0;i<n;i++)
        {
            if(firstIndex == -1 && arr[i] == target)
            {
                firstIndex = i;
            }
            if(arr[i] == target)
            {
                secondIndex = i;
            }
        }

        System.out.println(firstIndex+" "+secondIndex);
    }
      public static void main (String[] args)
      {
            //your code here

            Scanner in = new Scanner(System.in);

            int n = in.nextInt();
            int target = in.nextInt();

            int arr[] = new int[n];

            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            FirstAndLastPosition(arr,n,target);
      }
}
```

13th (First and Last position using functions - 2)

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

public class Main
{
    public static void FirstAndLastPosition(int arr[],int n,int target)
    {
        int firstIndex = -1;
        int secondIndex = -1;

        for(int i=0;i<n;i++)
        {
            if(arr[i] == target)
            {
                firstIndex = i;
                break;
            }
        }

        for(int i=(n-1);i>=0;i--)
        {
            if(arr[i] == target)
            {
                secondIndex = i;
                break;
            }
        }

        System.out.println(firstIndex+" "+secondIndex);
    }
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here

            Scanner in = new Scanner(System.in);

            int n = in.nextInt();
            int target = in.nextInt();

            int arr[] = new int[n];

            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            FirstAndLastPosition(arr,n,target);
      }
}
```

13th (NcR question Functions)

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

```java
import java.lang.*;
import java.io.*;

public class Main
{
    public static long fact(int n)
    {
        long fact = 1;
        for(int i=1;i<=n;i++)
        {
            fact = fact*i;
        }

        return fact;
    }
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);

        int n = in.nextInt();
        int r = in.nextInt();

        long num = fact(n);
        long deno = fact(n-r);

        long ans = num/deno;
        ans = ans/fact(r);

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

13th (Spiral Matrix)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
```

```
    {
        //your code here

        Scanner in = new Scanner(System.in);
        int r = in.nextInt();
        int c = in.nextInt();

        int arr[][] = new int[r][c];

        for(int i=0;i<r;i++)
        {
            for(int j=0;j<c;j++)
            {
                arr[i][j] = in.nextInt();
            }
        }

        int left = 0; // first column index
        int right = c-1; // last column index
        int top = 0; // first row index
        int bottom = r-1; // last row index

        while(left<=right && top<=bottom) // stopping check
        {
            // 1st arrow
            for(int i=left;i<=right;i++)
            {
                System.out.print(arr[top][i]+" ");
            }

            boolean flag = false;

            // 2nd arrow
            top++;
            for(int i=top;i<=bottom;i++)
            {
                flag = true;
                System.out.print(arr[i][right]+" ");
            }

            if(flag == true)
            {
                flag = false;
```

```
                    // 3rd arrow
                right--;
                for(int i=right;i>=left;i--)
                {
                     flag = true;
                     System.out.print(arr[bottom][i]+" ");
                }
                }


                if(flag == true)
                {
                    // 4th arrow
                bottom--;
                for(int i=bottom;i>=top;i--)
                {
                    System.out.print(arr[i][left]+" ");
                }
                }


                left++;
            }
      }
}
```

14th Oct (Time Complexity - 1)

```
public class Main
{
      public static void main(String[] args)
      {

            // 1. Number of operations in one second -> 10^8
            // 10^8 --> 1
        //   (multiplying by 10)
            // 10^9 --> 10 seconds

            // O(1) --> Constant Time Complexity
            // O(log n) --> logamathic time Complexity
            // O(N) --> Linear
            // O(Nlog N) (base 10)
            // O(N^2) --> Quadratic time complexity
            // O(N^3)


            Arrays.sort(); // Merge sort --> Nlog N

            // Big O notation

            int n = in.nextInt(); // O(1) // 1 <= N <= 10^6
            int m = in.nextInt();


            // Operations - N
            // Time Complexity - O(N)

            // Best case --> minimum number of operations in that scenario
            // Worst Case --> Maximum number of Operations (N is very high)
            for(int i=0;i<n;i++)
            {

            }

            // Operations - N*M
            // Time Complexity - O(N*M)
            for(int i=0;i<n;i++)
            {
                for(int j=0;j<m;j++)
                {

                }
            }

            // Operations = log n (base 2)
            // Time Complexity = O(log n)
            for(int i=1;i<n;i=i*2) // (i = 1,2,4,8,16,32_____ --> log n)
            {

            }

            // Operations = N^3
            // Time Complexity = O(N^3)
            for(int i=0;i<n;i++)
            {
```

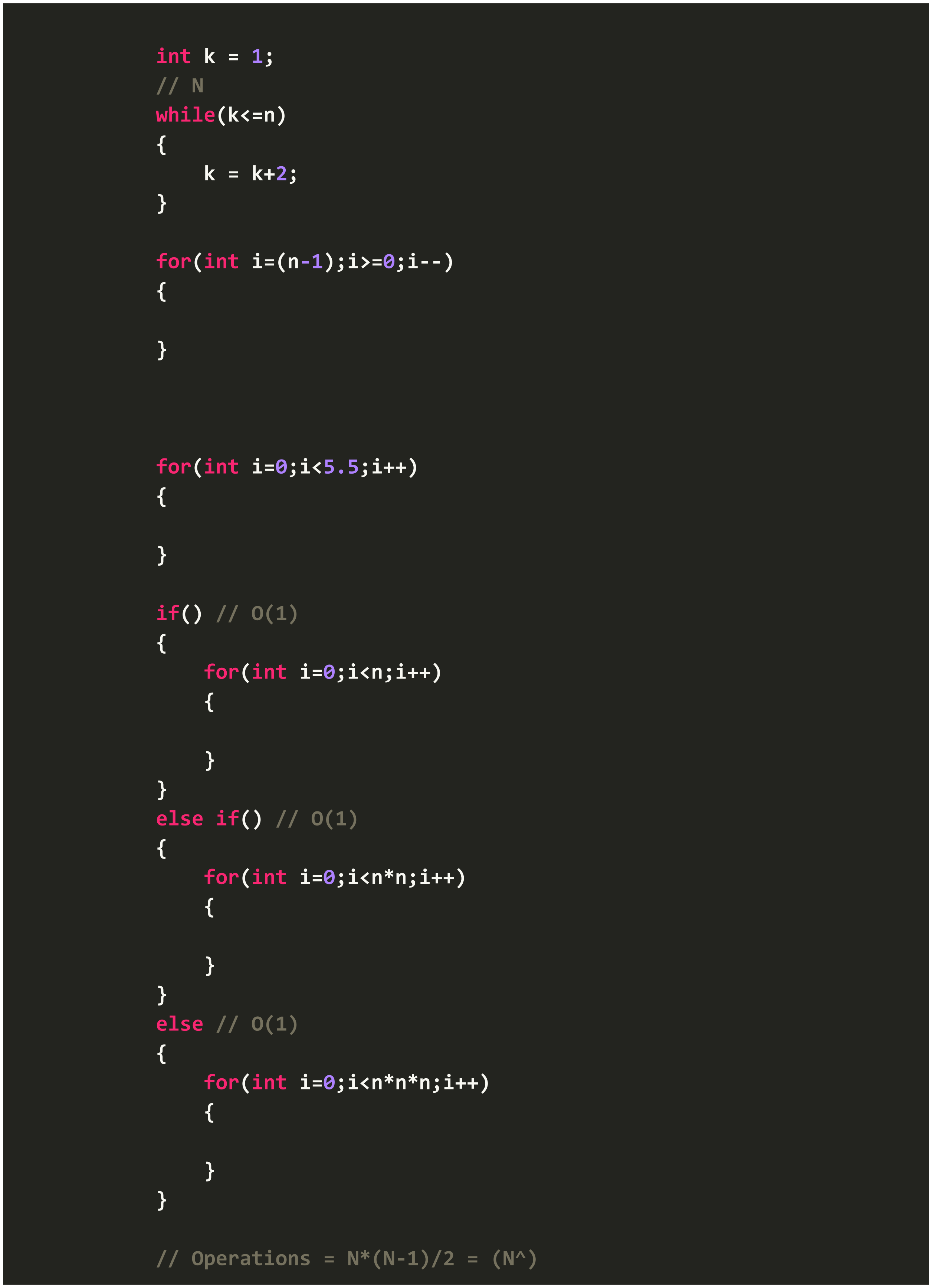
```
                for(int j=0;j<n*n;j++)
                {

                }
            }

            // Operations = (N^2+N) (N^2 >>>> N)
            // Time Complexity - O(N^2)
            for(int i=0;i<n;i++)
            {

            }
            for(int j=0;j<n*n;j++)
          {

          }


            // Operations = (n/2)
            // Time Complexity = O(n) (It will not be N/2 ? constants are 
ignored)
            for(int i=1;i<=n;i=i+2) // (i = 1,2,4,8,16,32_____ --> log n)
            {

            }

            // Operations - 10*n
            // Time Complexity - O(N) (We are ignoring the constants)
            for(int i=0;i<10*n;i++)
            {

            }

            // Operations - 100*n
            // Time Complexity - O(N*M (where M is 100))
            for(int i=0;i<100*n;i++)
            {

            }

            // Operations - sqrt(N)
            // Time Complexity - O(sqrt(N))
            for(int i=0;i<Math.sqrt(n);i++)
```

```java
{

}


int a = 5; // 1 unit, O(1)
System.out.println(); // 1 unit, O(1)

// 100 unit
for(int i=0;i<100;i++)
{
    // anything
}

// 10^6 operations
for(int i=0;i<1000;i++)
{
    int a = 5; // 1
    int b = 7; // 1
    int c = 7; // 1
    for(int j=0;j<1000;j++) // 1000
{

}
}

// Lesser the number of operations, better will be the program

// 10^3 operations
for(int i=0;i<1000;i++)
{
    int a = 5;
    int b = 7;
    int c = 7;

}

int k=0;
// 100 operations
while(k<100)
{
    // anything
    k++;
```

```
            }
        }
}
```

14th Oct (Time Complexity - 2)

```
public class Main
{
        public static void main(String[] args)
        {

                // 1. Number of operations in one second -> 10^8
                // 10^8 --> 1
            //   (multiplying by 10)
                // 10^9 --> 10 seconds

                // O(1) --> Constant Time Complexity
                // O(log n) --> logamathic time Complexity
                // O(sqrt(N))
                // O(N) --> Linear
                // O(Nlog N) (base 10)
                // O(N^2) --> Quadratic time complexity
                // O(N^3)

                // O(2^N) --> Exponential Time Complexity
                // O(N^N)

                int n = 1000;
                int m = 1000;

                for(int i=0;i<(n+5);i++) // Time Complexity - O(N)
                {

                }

                int k = n;
                // log n
                while(k>=1)
                {
                    k = k/n;
                }
```



```
int k = 1;
// N
while(k<=n)
{
    k = k+2;
}

for(int i=(n-1);i>=0;i--)
{

}



for(int i=0;i<5.5;i++)
{

}

if() // O(1)
{
    for(int i=0;i<n;i++)
    {

    }
}
else if() // O(1)
{
    for(int i=0;i<n*n;i++)
    {

    }
}
else // O(1)
{
    for(int i=0;i<n*n*n;i++)
    {

    }
}

// Operations = N*(N-1)/2 = (N^)
```

```
  // Time Complexity = O(N^2)

  // i = 0, j --> 0
// i = 1, j --> 1
// i = (n-1), j--> (n-1)

// Operations = 0+1+2+3+4 ----- (n-1)
  for(int i=0;i<n;i++)
  {
      for(int j=0;j<i;j++)
  {

  }
  }

  // Operations = n + nlog n
  // Time Complexity = O(Nlog n)
  for(int i=1;i<=n;i++)
  {
      // i = 1, LOOP 2 --> n
      // i = 2, LOOP 2 --> (n/2)
      // i = 3, LOOP 2 --> (n/3)
      // i = 4, LOOP 2 --> (n/4)

      // i = n,  LOOP 2 --> 1
      for(int j=1;j<=n;j=j+i) // LOOP 2
  {

  }
  }

  // Time Complexity - n * (n log n) = n^2 * logn
  for(int i=0;i<n;i++)
  {
      Arrays.sort(); // n log n
  }

  // Operations = N*N/2
  // Time Complexity = N^2
  for(int i=0;i<n;i++)
  {
      for(int i=0;j<n;j++)
      {
          if(j==(n/2))
          {
              break;
          }
      }
  }


  // Operations = N/2+1
  // Time Complexity = N
  for(int i=0;i<n;i++)
  {
      for(int i=0;j<n;j++)
      {
          if(j==(n/2))
          {
              break;
          }
      }
      break;
  }
  Arrays.sort(); // n log n

  // Operations = (N^2/4 * log n)
  // Time Complexity = O(N^2 * log n)
  for (int i = 0; i < n / 2; i++) // (N/2) , N
  {
  for (int j = 1; j + n / 2 <= n; j++) // (N/2), N
  {
      for (int k = 1; k <= n; k = k * 2) // log n, log n
      {

      }
  }
  }

  // Operations = log n
  // Time Complexity = log n
  for(int i=n;i>=1;i=i/2) // logn
  {

  }
```

```
//          Time Complexity - O(N^3)
      }
}
```

14th Oct (Space Complexity)

```
public class Main
{
      public static void main(String[] args)
      {
            // Stack, Queue, Linked list, Array, ArrayList, Hashmap, Matrix

            // O(1) --> Constant Space Complexity
            // O(log n) --> logamathic Space Complexity
            // O(N) --> Linear
            // O(Nlog N) (base 10)
            // O(N^2) --> Quadratic Space complexity
            // O(N^3)


            // Maximum number of elements you can store ~ 10^7
            // 10^5 / 10^6

//          1 <= n <= 100

            int n = in.nextInt(); // O(1)
            int m = in.nextInt(); // O(1)

            // 300 ~ O(1)

            int fre[] = new int[256]; // ~ O(1)

            int arr[] = new int[n*n]; // O(N)
            int brr[] = new int[n];

            int a[][] = new int[n][m]; // O(N*M) // (N =10^4 M = 10^4)

            // Operations = sqrt(N)
            // Time complexity = O(sqrt(N))
            for(int i=0;i*i<n;i++)
            {

            }
      }
}
```

16th Oct (Contest Matrix Question)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[][] = new int[n][n];


        for(int i = 0;i<n;i++)
        {
                for(int j = 0;j<n;j++)
                {
                        arr[i][j] = sc.nextInt();
                }
        }

        for(int i = 0;i<n;i++)
        {
            int col = 0;
            int row = 0;
                for(int j = 0;j<n;j++)
                {
                        row += arr[i][j];
                        col+=arr[j][i];
                }
                System.out.print((col - row)+" ");

        }
      }
}
```

16th Oct (String rotation)

```
import java.util.*;
public class Main
{
      public static void main(String[] args) {
          Scanner in = new Scanner(System.in);

          String a = in.next();
          String b = in.next();

          int l = b.length();
          a = a+a;

          boolean flag = false;

          for(int i=0;i+l<a.length();i++) // A.length()
          {
            String currentWindow = a.substring(i,i+l); // O(L) L =
B.length() m
            if(currentWindow.compareTo(b) == 0)
            {
                flag = true;
                break;
            }
          }

          // Time complexity = n*m

          if(flag == true)
          {
              System.out.println("YES");
          }
          else
          {
              System.out.println("NO");
          }

      }
}
```

16th Oct (Minimum length Word - Approach 1)

```
import java.util.*;
public class Main
{
      public static void main(String[] args) {
          Scanner in = new Scanner(System.in);

          String a = in.nextLine();

          int i = 0;
          int j = 0;

          int minimumLength = Integer.MAX_VALUE;

          String ans = ""; // empty string

          int l = a.length();
          while(j<l)
          {
              // keep incrementing j till when as soon as we find space or
sentence is over
              while(j<l && a.charAt(j)!=' ')
              {
                  j++;
              }

              int lengthOfWord = (j-i);
              if(lengthOfWord < minimumLength)
              {
                  minimumLength = lengthOfWord;
                  ans = a.substring(i,j);
              }

              j++;
              i = j;
          }
```

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

16th Oct (Minimum length Word - Approach 2)

```
import java.util.*;
public class Main
{
	public static void main(String[] args) {
	    Scanner in = new Scanner(System.in);
	    
	    String a = in.nextLine();
	    
	    int i = 0;
	    int j = 0;
	    
	    int minimumLength = Integer.MAX_VALUE;
	    
	    int indexOfWord = -1;
	    
	    int l = a.length();
	    while(j<l && i<l)
	    {
	        // keep incrementing j till when as soon as we find space or 
sentence is over
	        while(j<l && a.charAt(j)!=' ')
	        {
	            j++;
	        }
	        
	        int lengthOfWord = (j-i);
	        if(lengthOfWord < minimumLength)
	        {
	            minimumLength = lengthOfWord;
	            indexOfWord = i;
	        }
```

```
              j++;
              i = j;
          }

          System.out.println(a.substring(indexOfWord,
indexOfWord+minimumLength));
      }
}
```

17th Oct (Bubble Sort)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

        // 1. We want n-1 elements to be present in it's correct position
(For loop 1)
        // 2. Using swapping largest element in box will come to it's
correct position (For loop 2)

        // Time Complexity = O(N^2)
        // Operations = (n*(n-1))/2
        // Space Complexity = O(1) (for bubble sort logic no extra space is
required)

        // Best Case (Array is sorted) --> O(N)
        // Worst Case (Array is des order) --> O(N^2)
        for(int i=0;i<(n-1);i++) // n = 5, n-1 = 4 (0,1,2,3)
        {
            // i =0, j--> n-1
            // i =1, j--> n-2
            // i =2, j--> n-3

            // i =(n-2), j--> 1

            // (n-1)+ (n-2)+(n-3).... 1

            boolean swap = false;

            for(int j=0;j<(n-i-1);j++) // j can be atmax --> n-2 (n=5)
            {
                // arr[j], arr[j+1]
                if(arr[j] > arr[j+1])
                {
                    // swap
                    swap = true;
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }

            if(swap == false) // there is no swapping in box, array is
already sorted
            {
                break;
            }
            // Largest element in box will definitely reach it's correct
position
        }

        for(int i=0;i<n;i++)
        {
            System.out.print(arr[i]+" ");
        }

      }
```

```
}
```

17th Oct (Bubble Sort Assignment Question)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

        // 1. We want n-1 elements to be present in it's correct position
(For loop 1)
        // 2. Using swapping largest element in box will come to it's
correct position (For loop 2)

        // Time Complexity = O(N^2)
        // Operations = (n*(n-1))/2
        // Space Complexity = O(1) (for bubble sort logic no extra space is
required)

        // Best Case (Array is sorted) --> O(N)
        // Worst Case (Array is des order) --> O(N^2)

        int numberOfSwapOperation = 0;

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

            // (n-1)+ (n-2)+(n-3).... 1

            boolean swap = false;

            for(int j=0;j<(n-i-1);j++) // j can be atmax --> n-2 (n=5)
            {
                // arr[j], arr[j+1]
                if(arr[j] > arr[j+1])
                {
                    // swap
                    numberOfSwapOperation++;
                    swap = true;
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }

            if(swap == false) // there is no swapping in box, array is
already sorted
            {
                break;
            }
            // Largest element in box will definitely reach it's correct
position
        }

        System.out.println("Array is sorted in " + numberOfSwapOperation +
" swaps.");
        System.out.println("First Element: "+arr[0]);
        System.out.println("Last Element: "+arr[n-1]);

      }
}
```

17th Oct (Selection Sort)

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

public class Main
{
    /// 12347846891234
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

        // 1. N-1 minimum element needs to placed at it's correct position
(For loop 1)
        // 2. Find index of minimum element and swap it (For loop and if
condition)

        // Operations = (n*(n-1))/2
        // Time Complexity = O(N^2)

        int totalSwaps = 0;
        // Best Case (Array is sorted) --> O(N^2)
        // Worst Case --> O(N^2)
        for(int i=0;i<(n-1);i++)
        {
            int minimumElement = arr[i];
            int indexOfMinimumElement = i;

            // i = 0, j = n-1
            // i = 1, j = n-2

            // i = n-1, j = 0

            // iterating on rest other elements
            for(int j=(i+1);j<n;j++)
            {
                if(arr[j] < minimumElement)
                {
                    minimumElement = arr[j];
                    indexOfMinimumElement = j;
                }
            }

            // swapping ith element & indexOfMinimumElement

            if( i != indexOfMinimumElement)
            {
                totalSwaps++;
                int temp = arr[i];
                arr[i] = arr[indexOfMinimumElement];
                arr[indexOfMinimumElement] = temp;
            }

        }


        for(int i=0;i<n;i++)
            {
                System.out.print(arr[i]+" ");
            }
       }
}
```

17th Oct (Frequency Array)

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

public class Main
{
    /// 12347846891234
       public static void main (String[] args) throws java.lang.Exception
       {
            //your code here
         Scanner in = new Scanner(System.in);
         int n = in.nextInt();

```

```java
            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            // Time Complexity - O(N)
            // Space Complexity - O(maximumElement)

        // 1. Find maximum value
        // 2. Create a frequency array of size = (maximumElement + 1)
        // 3. We will calculate frequency of every element

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

        // indexes behave as original elements of the array

        // initially all elements have 0 frequency
        int fre[] = new int[(maximumElement+1)];

        // array -> 5 2 3 4 5
        // frequency array -> 0 0 0 0 0 0

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

        for(int i=0;i<=maximumElement;i++)
        {
            if(fre[i]!=0)
            {
                System.out.println(i+" --> " + fre[i]);
            }
```

```
         }
       }
}
```

18th Oct (Insertion Sort)

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

public class Main
{
      public static void main(String[] args)
      {
            // Insertion Sort
            // Sorting Question
            // ArrayList
            // Lambda Function

            // N = 10^5 (merge sort)


//          Arrays.sort(); // nlog n
            Scanner in = new Scanner(System.in);
            int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            // Time Complexity in worst case - O(N^2)
            // Operations in worst case - (n*(n-1))/2

            // Operations in best case = N-1
            // Time Complexity in best case - O(N)

            // Space Complexity - O(1)

            // 1. We try to increase the size of the box
            // 2. Last inserted element in the box will be placed in
correct position
            // 3. All the other elements will be shifted

            for(int i=1;i<n;i++) // (N-1)
            {
                int currentElement = arr[i]; // this element I want to
insert to it's correct position
                int j = i - 1;

                // for(int j=(i-1);j>=0 && arr[j]>currentElement; j--)

                while(j>=0 && arr[j]>currentElement)
                {
                    // arr[j] is representing the greater element
                    // shift the greater element
                    arr[j+1] = arr[j];
                    j--;
                }

                // j+1 will be the correct position
                arr[j+1] = currentElement;
            }

            for(int i=0;i<n;i++)
            {
                System.out.print(arr[i]+" ");
            }

      }
}
```

18th Oct (Maximum of 3 largest Element)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);
            int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

        Arrays.sort(arr); // ascending

        long ans1 = arr[n-1]*arr[n-2]*arr[n-3];
        long ans2 = arr[0]*arr[1]*arr[n-1];

        if(ans1 > ans2)
        {
            System.out.print(ans1);
        }
        else
        {
            System.out.print(ans2);
        }

      }
}
```

19th Oct (Recursion - 1)

```
public class Main
{
    // print all numbers from 1 to N
    public static void print(int currentNumber,int n)
    {
        // base case
        if(currentNumber>n)
        {
            return;
        }

        // recursive case
        // System.out.print(currentNumber+" ");  // Print and then make a
```

```
call
        print((currentNumber+1), n);

        System.out.print(currentNumber+" "); // call is made before then
you are printing
    }

    // sum of all numbers from 1 to N
    public static int recur(int currentNumber,int n)
    {
        // base case
        if(currentNumber > n)
        {
            return 0;
        }

        // recursive case
        int currentAns = currentNumber + recur(currentNumber+1,n);
        // System.out.println(currentAns);
        return currentAns;
    }

    // sum of all numbers from 1 to N (in reverse order)
    public static int recur1(int n)
    {
        // base case
        if(n == 0)
        {
            return 0;
        }

        // recursive case
        int currentAns = n + recur1(n-1);
        return currentAns;
    }
      public static void main(String[] args) {

            int n = 3;
//          print(1,n);
            int ans = recur(1,n);

            System.out.println(ans);
            // base case --> stopping condition (To stop the recursion)
            // recursive case --> any line any function calls itself, it
will be recursive case

            //Recursion --> When a function calls itself then we will say
that function recursive function
      }
}
```

19th Oct (Hello Recursion)

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

public class Main
{
    public static int recur(int index,int arr[],int n)
    {
        // base case
        if(index == n)
        {
            return 0;
        }

        // if(index == (n-1))
        // {
        //     return arr[index];
        // }

        // recursive case
        int currentAns = arr[index] + recur(index+1,arr,n);
        return currentAns;
    }
      public static void main (String[] args)
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int t = in.nextInt();
            int testCase = 1;
            while(t>0)
            {
                int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            int ans = recur(0,arr,n);
            System.out.println("Case " + testCase + ": " + ans);

            testCase++;
            t--;
            }
      }
}
```

20th Oct (Fibonacci Number)

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

public class Main
{
    // Nth number of fibonacci using recursion
    // F(N) = F(N-1) + F(N-2)
    // F(1) = 0, F(2) = 1

    // If something has been calculated, using DP (Dynamic Programming) (we
will not recalculate it)
    // Time Complexity - O(2^N)
    // All function call are stored in stack
    // some Extra space has been used by stack for storing function calls
of recursion --> O(2^N)
    public static long fibo(int n)
    {
        // base case
        if(n==1)
        {
            return 0;
        }
        if(n==2)
        {
            return 1;
        }

        // recursive case
        long nthNumber = fibo(n-1) + fibo(n-2);
        return nthNumber;
    }
      public static void main (String[] args)
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();
            long ans = fibo(n);
            System.out.println(ans);
      }
}
```

20th Oct (noX Question Recursion)

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

public class Main
{
    public static String recur(int index,String s)
    {
        // base case
        if(index >= s.length())
        {
            return ""; // "" --> Empty string
        }

        // recursive case
        if(s.charAt(index) == 'x')
        {
            // ignore the character at this index, answer is coming from
next recursion call
```

```
            return recur(index+1,s);
        }
        else
        {
            // Final Ans = this character + String from recursion
            String currentAns = s.charAt(index) + recur(index+1, s);
            return currentAns;
        }
    }
      public static void main (String[] args)
      {
            //your code here
            Scanner in = new Scanner(System.in);

            String s = in.next();
            String ans = recur(0,s);
            System.out.print(ans);
      }
}
```

20th Oct (Recursive Digit Sum)

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

public class Main
{
    public static int recur(int n)
    {
        // base case
        if(n<10)
        {
            return n;
        }

        // recursive case
        int sum = 0;
        while(n>0)
        {
            int digit = n%10;
            sum += digit;
            n = n/10;
        }

        return recur(sum);
    }
      public static void main (String[] args)
      {
            //your code here
            Scanner in = new Scanner(System.in);

            String s = in.next();
            int k = in.nextInt();

            int sumOfDigits = 0;
            for(int i=0;i<s.length();i++)
            {
                sumOfDigits += (s.charAt(i)-'0'); // '4' - '0' = 4
            }

            sumOfDigits = sumOfDigits*k; // k times we needed to
concatenate

            int ans = recur(sumOfDigits);
            System.out.print(ans);
      }
}
```

23rd Oct (Contest Question Discount from Shop)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

            int n = in.nextInt();

            int arr[] = new int[n];
            for(int i=0;i<n;i++)
            {
                arr[i] = in.nextInt();
            }

            Arrays.sort(arr);
            int cost = 0;
            int i = n-1;
            while(i>=0)
            {
                // adding two values
                cost += arr[i];
                if(i>=1) // i becomes 0
                {
                    cost += arr[i-1];
                }

                // decrementing the value of i
                i = i-3;
            }

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

23rd Oct (Contest Question Discount from Shop - 2)

```
import java.util.Scanner;
import java.util.Arrays;
class Solution{
    public int minimumCost(int[] arr, int n) {
        Arrays.sort(arr);
        int cost = 0;
            int i = n-1;
            while(i>=0)
            {
                // adding two values
                cost += arr[i];
                if(i>=1) // i becomes 0
                {
                    cost += arr[i-1];
                }

                // decrementing the value of i
                i = i-3;
            }

            return cost;
    }
}
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int[] cost = new int[n];
        for(int i=0; i<n; i++)  {
            cost[i]=sc.nextInt();
        }
        Solution obj=new Solution();
        int ans= obj.minimumCost(cost, n);
        System.out.print(ans);
    }
}
```

23rd Oct (Minimum Sum After Dividing the number)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);
            int a = in.nextInt();
            String s = Integer.toString(a); // "2932"
            // parseInt() --> String to Integer
            // toString() --> Integer to String

            char arr[] = new char[(s.length())];
            for(int i=0;i<4;i++)
            {
                arr[i] = s.charAt(i);
            }
            // ['2','9','3','2']

            Arrays.sort(arr);
            // ['2', '2', '3', '9']

            int ans = (arr[0]-'0')*10+(arr[3]-'0') +
(arr[1]-'0')*10+(arr[2]-'0');
            System.out.print(ans);

      }
}
```

23rd Oct (Maximum Subarray Sum - Kadane's Algo)

```
class Solution {
    //
    public int maxSubArray(int[] nums) {
        // kadane's Algo

        int currentSumEnding = 0;
        int maxSum = Integer.MIN_VALUE;

        int n = nums.length;

        for(int i=0;i<n;i++)
        {
            int firstOption = arr[i];
            int secondOption = arr[i]+currentSumEnding;

            currentSumEnding = Math.max(firstOption, secondOption);
            maxSum = Math.max(maxSum, currentSumEnding);
        }

        return maxSum;
    }
}
```

23rd Oct (Integer to Char)

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

public class Main
{
	public static void main (String[] args)
	{
	    // int --> char use type casting
	    // char --> int (No type casting needed)
	    char x = 'd';
	    System.out.println(x);

	    char y = (char)(x-32); // line requires type casting
	    char w = ('d'-32); // line works without type casting

	    int z = 'd';
	    int a = x;

	    System.out.println(z);
	    System.out.println(a);
	}
}
```

23rd Oct (Contest Question - Toggle Question)

```java
import java.util.*;

class Solution {
    static String Toggle(String str) {
       // Your code here

       int l = str.length();
       char ans[] = new char[l];

       // int --> char

       for(int i=0;i<l;i++)
       {
           if(str.charAt(i) >= 'A' && str.charAt(i) <= 'Z')
           {
               ans[i] = (char)(str.charAt(i)-'A'+'a');
           }
           else if(str.charAt(i) >= 'a' && str.charAt(i) <= 'z')
           {
               ans[i] = (char)(str.charAt(i)-'a'+'A');
           }
           else
           {
               ans[i] = str.charAt(i);
           }
       }

       String finalAnswer = new String(ans);
       return finalAnswer;
    }
}

public class Main {
    public static void main(String args[]) {
        String str;
        Scanner sc = new Scanner(System.in);
        str = sc.nextLine();
        Solution Obj = new Solution();
        System.out.print(Obj.Toggle(str));
    }
}
```

23rd Oct (Count ABC)

```
 //Count ABC
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{

        public static int countABC(String str,int i)
        {
                // base case
                    if(i+3>str.length())
                    {
```

```
                              return  0;
                      }

                //recursive call
                if(str.substring(i,i+3).compareTo("abc")==0 ||
str.substring(i,i+3).compareTo("aba") == 0)
                {

                        return 1+countABC(str,i+1);

                }
                else
                {

                        return countABC(str,i+1);

                }
        }
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
                Scanner in=new Scanner(System.in);
                String s=in.next();

                 int  ans=  countABC(s,0);
                System.out.println(ans);
      }
}
```

27th Oct (ArrayList)

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

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

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

```
 ArrayList<Integer> a = new ArrayList<>();
 for(int i=0;i<n;i++)
 {
     // a.get(i) = in.nextInt(); WRONG
     int x = in.nextInt();
     a.add(x);
 }

 int arr[] = new int[10];
 // arr.length

// ArrayList, Stack, Queue, Deque
//ArrayList<datatype> arrayListName = new ArrayList<>();

ArrayList<Long> b = new ArrayList<>();
ArrayList<Character> c = new ArrayList<>(); // char
ArrayList<Double> d = new ArrayList<>();
ArrayList<String> e = new ArrayList<>();

int z = 10;
// new elements added are stored in the last
a.add(z); // a = [10] // O(1)
a.add(5); // a = [10, 5]
a.add(2); // a = [10, 5, 2]
a.remove(1); // a = [10, 2]
a.set(0,100); // a = [100, 2]
a.remove(0); // a = [2]
a.remove(0); // a = [] (EMPTY)


// size() --> how many elements are present in arraylist O(1)
// add() --> add elements into the arraylist  O(1)
// get(i) --> return ith element in arraylist  O(1)

// a.get(0) = 100; WRONG

// remove(index) --> removing the element at that index
// set(index, value) --> update the value present at that index

//for(int i=0;i<a.size();i++)
//{
//    // arr[i] --> ith element in an array
//    System.out.println(a.get(i)+" "); // O(1)
```

```
        //    // a.get(0) --> 10
        //    // a.get(1) --> 5
        //    // a.get(2) --> 2
        //}

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

        while(in.hasNextInt())
        {
            int x = in.nextInt();
            t.add(x);
        }

        for(int i=0;i<t.size();i++)
        {
            // arr[i] --> ith element in an array
            System.out.print(t.get(i)+" ");
            // a.get(0) --> 10
            // a.get(1) --> 5
            // a.get(2) --> 2
        }
      }
}
```

27th Oct (Merge Sort)

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

public class Main
{
    // merge (Combination part)
    public static void merge(int start,int mid,int end,int arr[])
    {
        // first half of the array --> (start -- mid)
        ArrayList<Integer> A = new ArrayList<>();
        for(int i=start;i<=mid;i++)
        {
            A.add(arr[i]);
        }
```

```
// second half of the array --> (mid+1 -- en)
ArrayList<Integer> B = new ArrayList<>();
for(int i=(mid+1);i<=end;i++)
{
    B.add(arr[i]);
}

// Combine these two sorted parts
ArrayList<Integer> C = new ArrayList<>();

int i = 0;
int j = 0;

while(i<A.size() && j<B.size())
{
    if(A.get(i) <= B.get(j))
    {
        C.add(A.get(i));
        i++;
    }
    else
    {
        C.add(B.get(j));
        j++;
    }
}

// Elements from B ArrayList are left
while(j<B.size())
{
    C.add(B.get(j));
    j++;
}

// Elements from A ArrayList are left
while(i<A.size())
{
    C.add(A.get(i));
    i++;
}

// C --> Sorted part of Two combined sorted arrays (0---)
```

```
        // Update the original Array (Some Segment of Array gets sorted)
        int v = start;
        for(int k=0;k<C.size();k++)
        {
            // what is the position in which element should be updated
            arr[v] = C.get(k);
            // arr[k] = C.get(k); WRONG
            // arr[start+k] = C.get(k); CORRECT
            v++;
        }
    }

    // divide --> log N
    public static void mergeSort(int start,int end,int arr[])
    {
        // base case
        if(start == end)
        {
            // there is only one element
            return;
        }

        // recursive case
        int mid = (start+end)/2;

        mergeSort(start,mid,arr); // first half of the array --> (st --
mid)
        mergeSort(mid+1,end,arr); // second half of the array --> (mid+1 --
en)

        // merge first & second half of the array
        merge(start,mid,end,arr); // --> O(N)
    }
      public static void main (String[] args)
      {
          Scanner in = new Scanner(System.in);

          int n = in.nextInt();
          int arr[] = new int[n];

          for(int i=0;i<n;i++)
          {
              arr[i] = in.nextInt();
          }

          // O(N log N)
          mergeSort(0,n-1,arr);

          for(int i=0;i<n;i++)
          {
              System.out.print(arr[i]+" ");
          }
      }
}
```

27th Oct (ArrayList of ArrayList)

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

public class Main
{
      public static void main (String[] args) throws java.lang.Exception
      {

//          ArrayList<datatype> arraylistName = new ArrayList<>();
            ArrayList<ArrayList<Integer>> arr = new ArrayList<>();

            ArrayList<Integer> a = new ArrayList<>();
            a.add(1);
            a.add(2);

            arr.add(a);
            a.add(3);
            arr.add(a);
      }
}
```

28th Oct (QuickSort)

```
import java.util.*;

public class Main
{
    // Modification of SORT --> QuickSelect (Leetcode Medium)

    // modify part --> Take last element as pivot, Arrange the elements and 
return the index of pivot element
    public static int partition(int start,int end,int arr[])
    {
        // Arrange the elements correct and after that return index of 
pivot element
        int pivot = arr[end];
        int i = start; // correct position where smaller values need to be 
placed
        for(int j=start;j<=(end-1);j++)
        {
            if(arr[j]<pivot)
            {
                // do something (place jth value at ith position)
                int tmp = arr[j];
                arr[j] = arr[i];
                arr[i] = tmp;

                i++; // so that next element can be at the correct position
            }
        }

        // pivot at ith position
        int tmp = arr[end];
        arr[end] = arr[i];
        arr[i] = tmp;

        return i; // return the index of pivot element
    }

    // recursive
    public static void quickSort(int start,int end,int arr[])
    {
        if(start > end)
        {
            return;
        }

        if(start == end)
        {
            return;
        }

        int indexOfPivot = partition(start,end,arr); // return the index of
pivot element

        // recursive case
        quickSort(start,indexOfPivot-1,arr); // First Part
        quickSort(indexOfPivot+1,end,arr); // Second Part
    }
    public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

          int n = in.nextInt();
          int arr[] = new int[n];

          for(int i=0;i<n;i++)
          {
              arr[i] = in.nextInt();
          }

          quickSort(0,n-1,arr);

          for(int i=0;i<n;i++)
          {
              System.out.print(arr[i]+" ");
          }
      }
}
```

28th Oct (Lambda Expression)

```
import java.util.*;

public class Main
{
    public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);

```

```
          ArrayList<Integer> arr = new ArrayList<>();

          int n = in.nextInt();
          for(int i=0;i<n;i++)
          {
              int x = in.nextInt();
              arr.add(x);
          }

          // Arrays.sort(array)
         // Collections.sort(arr, (a,b) -> (a>b ? positiveNumber:
negativeNumber));
         // positiveNumber --> swapping taking place
         // negativeNumber --> no swapping
          Collections.sort(arr, (x,y) -> (x>y ? 1:-1)); // new function
using this we can sort the array list

          for(int i=0;i<n;i++)
          {
              System.out.print(arr.get(i)+" ");
          }

      }
}
```

28th Oct (Largest Number Leetcode)

```
class Solution {
    public String largestNumber(int[] nums) {
        ArrayList<String> arr = new ArrayList<String>();
        int n = nums.length;

        for(int i=0;i<n;i++)
        {
            arr.add(String.valueOf(nums[i]));
        }

        Collections.sort(arr, (a, b)-> {
            String s1 = a + b;
            String s2 = b + a;
            return s2.compareTo(s1);
        });

        String ans = "";
        for(int i=0;i<n;i++)
        {
            ans+=arr.get(i);
        }

        if(ans.charAt(0) == '0')
        {
            return "0";
        }

        return ans;
    }
}
```

29th Oct (Encryption/ Decryption Question Interview)

```
import java.util.*;
public class Main
{
    public static String func(String s,int k)
    {
        int l = s.length();
            char a[] = new char[l];
            for(int i=0;i<l;i++)
            {
                a[i] = s.charAt(i);
            }

            for(int i=0;i<l;i++)
            {
                int value = (s.charAt(i) - 'a'); // 'a' --> 0, 'b' --> 1,
'c' --> 2
                // 0 - 25
                value = value+k; // value can be > 25
                value = value%26; // value --> 1 + 97 = 98 --> 'b'
                a[i] = (char)(value+'a');
                //  System.out.print(a[i]);
                // ans = ans+a[i] or ans = ans+s.charAt(i)
            }

            String ans = new String(a);
            return ans;
    }
      public static void main(String[] args) {
            // Module Test --> 60%
            // 5 mins --> 15 mins --> 20-25 mins

            // strings questions

            Scanner in = new Scanner(System.in);
            String s = in.next();
            int k = in.nextInt();

            String ans = func(s,k);
            System.out.print(ans);
      }
}
```

29th Oct (Frequency In sorted Array)

```
import java.util.*;
public class Main
{
      public static void main(String[] args) {
            // Module Test --> 60%
            // 5 mins --> 15 mins --> 20-25 mins

            Scanner in = new Scanner(System.in);

          int n = in.nextInt();
          int arr[] = new int[n];

          for(int i=0;i<n;i++)
          {
              arr[i] = in.nextInt();
          }

          // In case of unsorted first you need to sort the array
          // SORTED ARRAY

        int i = 0;
        int j = 0;
        while(j<n)
        {
            // increment the value of j until and unless, you get a
different element
            while(j<n && arr[j] == arr[i])
            {
                j++;
            }

            int fre = (j-i); // frequency of ith element (Element at which
i is pointing)
            System.out.println(arr[i]+" --> "+fre);

            i = j; // i is jumping on unique elements
        }
      }
}
```

29th Oct (Reverse words leetcode)

```
class Solution {
    public String reverseWords(String s) {

        int l = s.length();
          int i = 0;
          int j = 0;

          ArrayList<String> words = new ArrayList<>();

          while(j<l)
          {
              // skipping the spaces
              while(i<l && j<l && s.charAt(i) == ' ' && s.charAt(j) == ' ')
              {
                  i++;
                  j++;
              }

              // definately now there some character (Finding the word)
              while(j<l && s.charAt(j)!=' ')
              {
                  j++;
              }

            if((j-i)>=1)
            {
                String currentWord = s.substring(i,j);
                // System.out.print(currentWord+",");
                words.add(currentWord);
            }

              i = j;
          }

        String ans = "";
        for(int k=words.size()-1; k>=0; k--)
          {
              ans = ans + words.get(k);
            if(k != 0)
            {
                ans = ans + " ";
            }
          }

        return ans;
    }
}
```

29th Oct (Two sum in sorted array)

```
class Solution {
    public int[] twoSum(int[] arr, int t) {

        int n = arr.length;

        int ans[] = new int[2];
        ans[0] = -1;
        ans[1] = -1;

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

        while(i<j)
        {
            int currentSum = arr[i]+arr[j];

            if(currentSum == t)
            {
                ans[0] = i+1;
                ans[1] = j+1;
                break;
            }
            else if(currentSum > t)
            {
                j--;
            }
            else
            {
                i++;
            }
        }

        return ans;
    }
}
```

29th Oct (Zigzag Question Leetcode)

```
class Solution {
    public String convert(String s, int numRows) {
        if(numRows==1)
        {
            return s;
        }
        ArrayList<ArrayList<Character>> arr = new ArrayList<>();

        for(int i=0;i<numRows;i++)
        {
            ArrayList<Character> a = new ArrayList<>();
            arr.add(a);
        }

        int index = 0; // rowNumber
        int change = 1;
        for(int i=0;i<s.length();i++)
        {
            arr.get(index).add(s.charAt(i));
            index += change;

            if(index==numRows)
            {
                index -= 2;
                change = -1;
            }
            if(index<0)
            {
                index+=2;
                change = 1;
            }
        }

        String ans = "";
        for(int i=0;i<numRows;i++)
        {
            for(int j=0;j<arr.get(i).size();j++)
            {
                char currentCharacter = arr.get(i).get(j);
                ans+=currentCharacter;
            }
        }

        return ans;
    }
}
```

29th Oct (Inversion Count)

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

public class Main
{
    public static int merge(int st,int mid,int en,int arr[])
    {
        // st -- mid (first part) (0 -- 2) (0 1 2)
        // mid+1 -- en (second part)
        int currentAns = 0;
        ArrayList<Integer> A = new ArrayList<>();

        for(int i=st;i<=mid;i++) // first part elements
        {
            A.add(arr[i]);
        }

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

        for(int i=(mid+1);i<=en;i++) // second part elements
        {
            B.add(arr[i]);
        }

        int i = 0;
        int j = 0;

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

        while(i<A.size() && j<B.size()) // either of first or second is
finished
        {
            if(A.get(i)<=B.get(j))
            {
                C.add(A.get(i));
                i++;
            }
            else
            {
                C.add(B.get(j));
                currentAns += (A.size()-i);
                j++;
            }
        }

        // when first part is not finished
        while(i<A.size())
        {
            C.add(A.get(i));
            i++;
        }
        // when second part is not finished
        while(j<B.size())
        {
            C.add(B.get(j));
            j++;
        }

        // Copy all elements from C to original array ?
        for(int k=0;k<C.size();k++)
        {
            arr[st+k] = C.get(k);
        }

        return currentAns;
    }

    public static int mergesort(int st,int en,int arr[])
    {
        // base case
        if(st==en)
        {
            // one element
            return 0;
        }

        // recursive case
        int mid = (st+en)/2;

        int ans = 0;
        // first part
        ans += mergesort(st,mid,arr);

        // second part
        ans += mergesort(mid+1,en,arr);

        // merge first and second part
        ans += merge(st,mid,en,arr);

        return ans;
    }
      public static void main (String[] args) throws java.lang.Exception
      {
            //your code here
            Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        while(t>0)
        {
          int n = in.nextInt();
          int arr[] = new int[n];

          for(int i=0;i<n;i++)
          {
              arr[i] = in.nextInt();
          }

          int inv = mergesort(0,n-1,arr);

          System.out.println(inv);
          t--;
        }
      }
}
```

29th Oct (Lambda Expression)

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

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


        ArrayList<ArrayList<Integer>> a = new ArrayList<>();
            for(int i=0;i<n;i++)
            {
              int phy = in.nextInt();
              int chem = in.nextInt();
              int math = in.nextInt();

              ArrayList<Integer> arr = new ArrayList<>();
              arr.add(phy);
              arr.add(chem);
              arr.add(math);
              a.add(arr);
            }

            // Conditon ? 1 : -1
            // custom sort function ()

            Collections.sort(a, (x,y) -> (x.get(2) > y.get(2)) ? 1: -1);


//          Collections.sort(a, (x,y) -> (x.get(0) == y.get(0)) ? 
((x.get(1) > y.get(1)) ? 1: -1) : ((x.get(0) > y.get(0) ? 1:-1)));

            for(int i=0;i<n;i++)
            {
                System.out.println(a.get(i).get(0)+" "+a.get(i).get(1)+" 
"+a.get(i).get(2));
            }


        // ArrayList<Integer> arr = new ArrayList<>();
        // for(int i=0;i<n;i++)
        // {
        //     int x = in.nextInt();
        //     arr.add(x);

        // }

        // // Arrays.sort(a) --> sort array

        // // What Two elements you are considering -> (Condition) ? 1 : -1
        // Collections.sort(arr, (x,y) -> (x<y) ? 1: -1); // sort arraylist

        // for(int i=0;i<n;i++)
        // {
        //     System.out.print(arr.get(i)+" ");
        // }
    }
}
```