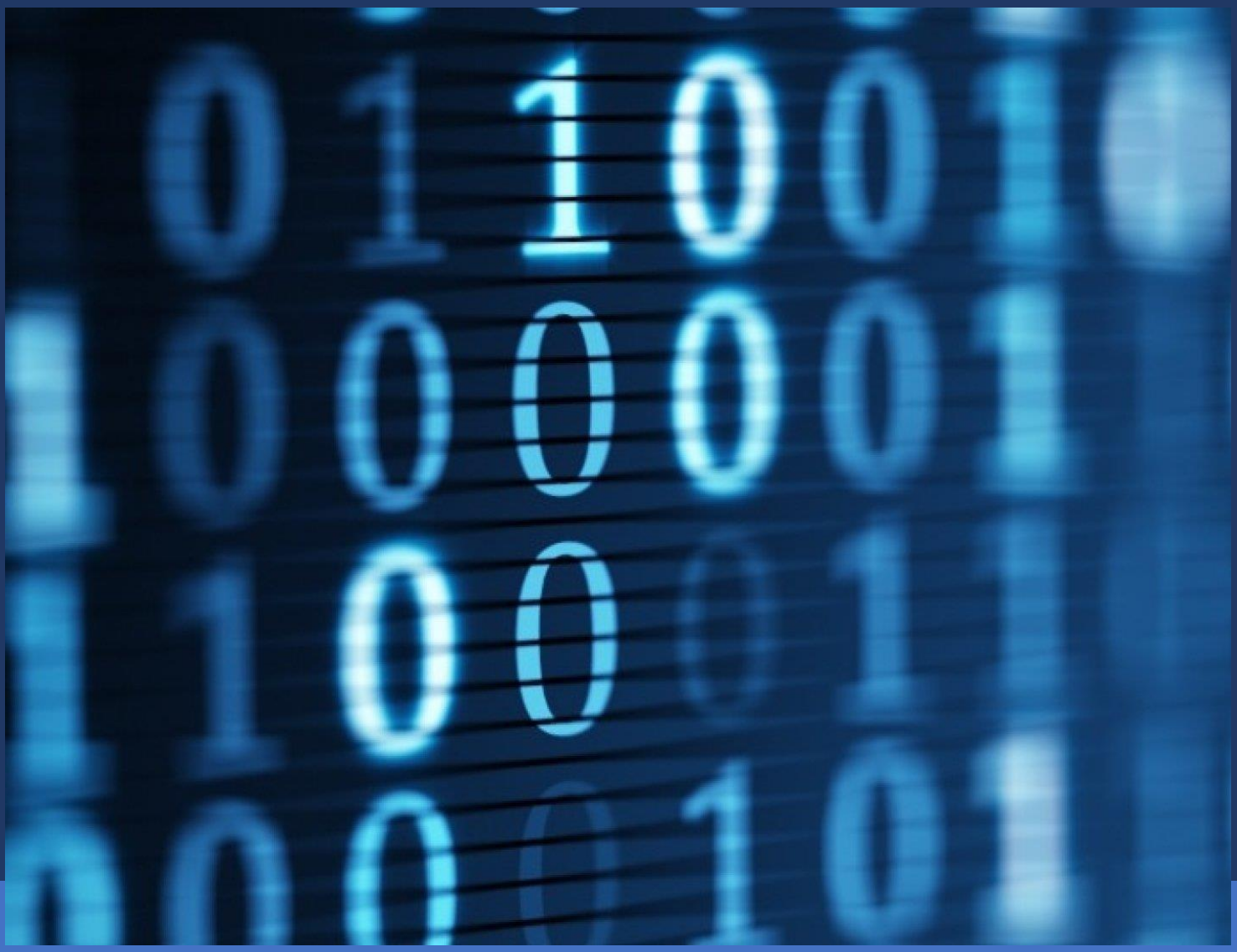

Computer Science

Programs pdf

Meemansha Sah

[Date]

[Course title]

INDEX

NUMBER THEORY

```
public class Number_Functions      //Functions to perform on numbers for different kinds of Numbers in number theory.
{
   public static void extract_digits(int N)
   {
      int C=N;
      while(C>0)
      {
         System.out.print(C%10+"     ");
         C/=10;
      }
   }
   public static int reverse(int N)
   {
      int C=N;
      int R=0;
      while(C>0)
      {
         R*=10;
         R+=C%10;
         C/=10;
      }
      return R;
   }
   public static int no_digits(int N)
   {
      int C=N; int c=0;
      while(C>0)
      {
         C/=10;
         c++;
      }
      return c;
   }
   public static int rotate_by1(int N)
   {
      int n=(int)Math.pow(10,(no_digits(N)-1));
      int R=N%n;
      R*=10;
      R+=N/n;
      return R;
   }
   public static int sumAllPrimeFactors(int N)
   {
      int C=N;int sum=0;int i=2;
      while(C>1)
      {
         if(C%i==0 & PrimeCheckC.isPrime(i)){ C/=i;sum+=i;}
         else i++;
      }
      return sum;
   }
   public static int sumDigits(int N)
   {
      int C=N;
      int sum=0;
```

```
        while(C>0)
        {
            sum+=C%10;
            C/=10;
        }
        return sum;
    }
    public static int sumSqDigits(int N)
    {
        int C=N;
        int sum=0;
        while(C>0)
        {
            int d=C%10;
            sum+=d*d;
            C/=10;
        }
        return sum;
    }
    public static int ProductDigits(int N)
    {
        int C=N;
        int pro=1;
        while(C>0)
        {
            pro*=C%10;
            C/=10;
        }
        return pro;
    }
    public static int sumOfFactors(int N)
    {
        int facSum=0;
        for(int i=1;i<=N/2;i++)
        {
            if(N%i==0) facSum+=i;
        }
        return facSum;
    }

    public static int[] allPrimeFactors(int N)
    {
        int C=N;int i=2;
        int f[]=new int[100];
        int j=0;
        while(C>1)
        {
            if(C%i==0 & PrimeCheckC.isPrime(i)){ C/=i;f[j]=i;j++;}
            else i++;
        }
        int F[]=TrimArray.trimAr(f,j);
        return F;
    }
}
```

```
public class Number_Checks
{
   public static boolean isPalindrome(int N)
   {
      if(N==Number_Functions.reverse(N)) return true;
      return false;
   }
   public static boolean isPerfect(int N)
   {
      int sum=0;
      for(int i=1;i<N;i++)
      {
         if(N%i==0) sum+=i;
      }
      if(sum==N) return true;
      else return false;
   }
   public static boolean isAdam_No(int N)
   {
      int rN=Number_Functions.reverse(N);
      if((rN*rN)==Number_Functions.reverse(N*N)) return true;
      return false;
   }
   public static boolean isSmith_No(int N)
   {
      int sumD=Number_Functions.sumDigits(N);
      int f[]=Number_Functions.allPrimeFactors(N);
      for(int i=0;i<f.length;i++)
      {
         if(f[i]>9) f[i]=Number_Functions.sumDigits(f[i]);
      }
      int sumF=0;
      for(int i=0;i<f.length;i++)
      {
         sumF+=f[i];
      }
      if(sumD==sumF) return true;
      return false;
   }
   public static boolean isArmstrong(int N)
   {
      int C=N;
      int sum=0;
      while(C>0)
      {
         int d=C%10;
         sum+= d*d*d;
         C/=10;
      }
      if(sum==N) return true;
      else return false;
   }
   public static boolean isNarcissistic(int N)
   {
      int C=N;
```

```
    int sum=0;
    int nd=Number_Functions.no_digits(N);
    while(C>0)
    {
        int d=C%10;
        sum+=Math.pow(d,nd);
        C/=10;
    }
    if(sum==N) return true;
    else return false;
}
public static boolean isDudeney(int N)
{
    int C=N;
    int sum=0;
    while(C>0)
    {
        sum+=C%10;
        C/=10;
    }
   int cu=sum*sum*sum;
   if(cu==N)return true; else return false;
}
public static boolean isMagic(int N)
{
    int C=N;
    int nd=Number_Functions.no_digits(N);
    while(nd>1)
    {
        C=Number_Functions.sumDigits(C);
        nd=Number_Functions.no_digits(N);
    }
    if (C==1) return true;
    return false;
}
public static boolean isAbundant(int N)
{
    int C=N;
    int facSum=0;
    for(int i=1;i<=C/2;i++)
    {
        if(C%i==0) facSum+=i;
    }
    if(facSum>N) return true;
    return false;
}
public static boolean isAutomorphic(int N)
{
    int sqN=N*N;
    if(sqN%10==N) return true;
    return false;
}
public static boolean isNivenNo(int N)
{
    int sum=Number_Functions.sumDigits(N);
```

```
        if(N%sum==0) return true;
        return false;
    }
    public static boolean isAmicable(int a)
    {
        int fSum1=0;
        int fSum2=0;
        for(int i=1;i<=a/2;i++)
        {
            if(a%i==0) fSum1+=i;
        }
        int b=fSum1;
        for(int i=1;i<=b/2;i++)
        {
            if(b%i==0) fSum2+=i;
        }
        if(fSum1==b && fSum2==a) return true;
        return false;
    }
    public static boolean isAmicable_pair(int a,int b)
    {
        int fSum1=0;
        int fSum2=0;
        for(int i=1;i<=a/2;i++)
        {
            if(a%i==0) fSum1+=i;
        }
        for(int i=1;i<=b/2;i++)
        {
            if(b%i==0) fSum2+=i;
        }
        if(fSum1==b && fSum2==a) return true;
        return false;
    }
}
```

```
class NumberRanges
{
   static void rangePalindrome(int l,int u)
   {
      System.out.println("Following are the Palindrome numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isPalindrome(i)) System.out.println("\t"+i);
      }
   }
   static void rangePerfect(int l,int u)
   {
      System.out.println("Following are the Perfect numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isPerfect(i)) System.out.println("\t"+i);
      }
   }
   static void rangeAdam(int l,int u)
   {
      System.out.println("Following are the Adam numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isAdam_No(i)) System.out.println("\t"+i);
      }
   }
   static void rangeSmith(int l,int u)
   {
      System.out.println("Following are the Smith numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isSmith_No(i)) System.out.println("\t"+i);
      }
   }
   static void rangeArmstrong(int l,int u)
   {
      System.out.println("Following are the Armstrong numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isArmstrong(i)) System.out.println("\t"+i);
      }
   }
   static void rangeNarcissistic(int l,int u)
   {
      System.out.println("Following are the Narcissistic numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(Number_Checks.isNarcissistic(i)) System.out.println("\t"+i);
      }
   }
   static void rangeDudeney(int l,int u)
   {
      System.out.println("Following are the Dudeney numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
```

```
        if(Number_Checks.isDudeney(i)) System.out.println("\t"+i);
      }
    }
    static void rangeMagic(int l,int u)
    {
      System.out.println("Following are the Magic numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
        if(Number_Checks.isMagic(i)) System.out.println("\t"+i);
      }
    }
    static void rangeAbundant(int l,int u)
    {
      System.out.println("Following are the Abundant numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
        if(Number_Checks.isAbundant(i)) System.out.println("\t"+i);
      }
    }
    static void rangeAutomorphic(int l,int u)
    {
      System.out.println("Following are the Automorphic numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
        if(Number_Checks.isAutomorphic(i)) System.out.println("\t"+i);
      }
    }
    static void rangeNivenNo(int l,int u)
    {
      System.out.println("Following are the NivenNo numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
        if(Number_Checks.isNivenNo(i)) System.out.println("\t"+i);
      }
    }
    static void rangeAmicable(int l,int u)
    {
      System.out.println("Following are the Amicable numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
        if(Number_Checks.isAmicable(i)) System.out.println("\t"+i);
      }
    }
}
```

```
class Prime_Operations
{
   public static boolean isPrime(int N)
   {
      int fc=0;
      for(int i=2;i<N;i++)
      {
         if(N%i==0) fc++;
      }
      if(fc==0) return true;
      else return false;
   }
   public static boolean isComposite(int N)
   {
      int fc=0;
      for(int i=2;i<N/2;i++)
      {
         if(N%i==0) fc++;
      }
      if(fc==0) return false;
      else return true;
   }
   public static boolean isSemiPrime(int N)
   {
      for(int i=2;i<=N/2;i++)
      {
         if(N%i==0 && isPrime(i) )
         {  int fac2=N/i;
            if(isPrime(fac2)) return true;
         }
      }
      return false;
   }
   public static boolean isInterPrime(int N)
   {
      if(isPrime(N-1) && isPrime(N+1)) return true;
      return false;
   }
   public static boolean isCircularPrime(int N)
   {
      int l=Number_Functions.no_digits(N);
      boolean R=true;
      for(int i=0;i<=l;i++)
      {
         N=Number_Functions.rotate_by1(N);
         if(isPrime(N)) R=true; else return false;
      }
      return R;
   }
   public static boolean isEmirpPrime(int N)
   {
      if(isPrime(N)&&isPrime(Number_Functions.reverse(N)))return true;
      return false;
   }
   public static boolean isPalindromicPrime(int N)
```

```
    {
        if(isPrime(N)&&Number_Checks.isPalindrome(N))return true;
        return false;
    }
    public static boolean isTwinPrime(int i)
    {
        if(isPrime(i)&&isPrime(i+2))
            return true;
        else return false;
    }
    public static boolean isCousinPrime(int i)
    {
        if(isPrime(i)&&isPrime(i+4))
            return true;
        else return false;
    }
    public static boolean isChenPrime(int i)
    {
        if(isPrime(i)&&(isPrime(i+2)||isSemiPrime(i+2)))
            return true;
        else return false;
    }
    public static boolean isSophieGermanPrime(int i)
    {
        if(isPrime(i)&&isPrime((2*i)+1))
            return true;
        else return false;
    }
    public static boolean isSafePrime(int i)
    {
        if(isPrime(i)&&isPrime((i-1)/2))
            return true;
        else return false;
    }
}
```

```
class PrimeRanges
{
   public static void primeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Prime numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isPrime(i)) System.out.print(" "+i+",");
      }
   }
   public static void compositeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Composite numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isComposite(i)) System.out.print(" "+i+",");
      }
   }
   public static void semiPrimeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Semi-Prime numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isSemiPrime(i)) System.out.print(" "+i+",");
      }
   }
   public static void circularPrimeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Circular-Prime numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isCircularPrime(i)) System.out.print(" "+i+",");
      }
   }
   public static void twinPrimeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Twin-Prime numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isTwinPrime(i)) System.out.print(" "+i+",");
      }
   }
   public static void cousinPrimeR(/*lower limit*/int l,/*upper limit*/int u)
   {
      System.out.println(" Cousin-Prime numbers between "+l+" and "+ u +" are: ");
      System.out.println(" ");
      for(int i=l;i<=u;i++)
      {
         if(Prime_Operations.isCousinPrime(i)) System.out.print(" "+i+",");
      }
```

```
}
public static void primeTripletsR(/*lower limit*/int l,/*upper limit*/int u)
{
   int pT[]=new int[3];
   System.out.println("Sets of Prime Triplets:");
      System.out.println(" ");
   for(int i=l;i<=u;i++)
   {
      int c=0;
      if(Prime_Operations.isPrime(i))
      {
         for(int j=0;j<=6;j+=2)
         {
            if(Prime_Operations.isPrime(i+j)){pT[c]=i+j; c++;if(c==3)break;}
         }
      }

      if(c>=3)
      {
         for(int j=0;j<3;j++)
         {
            System.out.print(pT[j]+" ");
         }
         System.out.println(" ");
      }
   }
}
public static void primeQuadrupletsR(/*lower limit*/int l,/*upper limit*/int u)
{
   int pT[]=new int[4];
   System.out.println("Sets of Prime Quadruplets:");
      System.out.println(" ");
      if(l<=3) l=4;
   for(int i=l;i<=u;i++)
   {
      int c=0;
      if(Prime_Operations.isPrime(i))
      {
         for(int j=0;j<=8;j++)
         {
            if(Prime_Operations.isPrime(i+j)){pT[c]=i+j; c++;if(c==4)break;}
         }
      }

      if(c>=4)
      {
         for(int j=0;j<4;j++)
         {
            System.out.print(pT[j]+" ");
         }
         System.out.println(" ");
      }
   }
}
public static void chenPrimeR(/*lower limit*/int l,/*upper limit*/int u)
```

```
    {
        System.out.println(" Chen-Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isChenPrime(i)) System.out.print(" "+i+",");
        }
    }
    public static void sophieGermanPrimeR(/*lower limit*/int l,/*upper limit*/int u)
    {
        System.out.println(" Sophie German Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isSophieGermanPrime(i)) System.out.print(" "+i+",");
        }
    }
    public static void safePrimeR(/*lower limit*/int l,/*upper limit*/int u)
    {
        System.out.println(" Safe-Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isSafePrime(i)) System.out.print(" "+i+",");
        }
    }
    public static void interPrimeR(/*lower limit*/int l,/*upper limit*/int u)
    {
        System.out.println(" Inter-Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isInterPrime(i)) System.out.print(" "+i+",");
        }
    }
    public static void emirpPrimeR(/*lower limit*/int l,/*upper limit*/int u)
    {
        System.out.println(" Emirp-Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isEmirpPrime(i)) System.out.print(" "+i+",");
        }
    }
    public static void palindromicPrimeR(/*lower limit*/int l,/*upper limit*/int u)
    {
        System.out.println(" Palindromic-Prime numbers between "+l+" and "+ u +" are: ");
        System.out.println(" ");
        for(int i=l;i<=u;i++)
        {
            if(Prime_Operations.isPalindromicPrime(i)) System.out.print(" "+i+",");
        }
    }
}
```

```
class Xylem_No
{
   public static boolean main(int n)
   {
      int c=n;
      int S1=n%10+n/(int)(Math.pow(10,Number_Functions.no_digits(n)-1));
      c/=10;
      int S2=0;
      while(c>10)
      {
         S2+=c%10;
         c/=10;
      }
      if(S1==S2) return true;
      else return false;
   }
}
```

```
class BouncyIncreaseDecrease
{
   int n;
   static boolean inc;
   static boolean dec;
   static boolean bouncy;
   BouncyIncreaseDecrease(int N)
   {
      n=N;
      inc=true;
      dec=true;
      bouncy=false;
   }

   void isIncreasingOrDecreasing()
   {
      int c=n;

      while(c>9)
      {
         int a=c%10;
         c/=10;
         int b=c%10;
         if(a<b) inc=false;
         if(a>b)dec=false;
      }
      if(inc==false && dec==false) bouncy=true;
   }
   public static void main(int I)
   {
```

```
      BouncyIncreaseDecrease N=new BouncyIncreaseDecrease(I);
      N.isIncreasingOrDecreasing();
      if(inc) System.out.println("The no. is increasing.");
      else if(dec) System.out.println("The no. is decreasing.");
      else if(bouncy) System.out.println("The no. is bouncy.");
   }
}

import java.util.Scanner;
class Tech_No
{
   public static void main(int N)
   {
      Scanner in=new Scanner(System.in);
      int n=Number_Functions.no_digits(N);
      if(n%2==1)
      {
         System.out.println("Invalid Input");
         System.out.println("Please input again");
         int a=in.nextInt();
         main(a);
         return;
      }
      double a=N%Math.pow(10,(n/2));
      double b=N/Math.pow(10,(n/2));
      if((a*a+b*b)==N)System.out.println("It is a tech no.");
      else System.out.println("It is not a tech no.");
   }
}









class Ramanujan_No
{
   public static void main(int N)
   {
      int C=N; int count=0;
      for(int i=1;i<=N;i++)
      {
         double b=C-i*i*i;
         double B=Math.cbrt(b);
         double d=B-(int)B;
         if(d==0)
         {
            count++;
         }
      }
      if(count>=3) System.out.println(N+" is a Ramanujan Number.");
      else System.out.println(N+" is not a Ramanujan Number.");
   }
}
```

```
class Karprekar_No
{
   static int extract(int N, int i)
   {
      int ex=0;
      ex+=N%Math.pow(10,i);
      return ex;
   }
   public static int no_digits(int N)
   {
      int C=N; int c=0;
      while(C>0)
      {
         int d=C%10;
         C/=10;
         c++;
      }
      return c;
   }
   static boolean isKarprekar(int N)
   {
      int n=N;
      int sq=N*N;
      int d=no_digits(N);
      int sum=0;
      sum+=extract( sq, d);
      n/=Math.pow(10,d);
      sum+=n;
      if(N==sum)return true;
      else return false;
   }
   static void rangeKarprekar(int l,int u)
   {
      System.out.println("Following are the Karprekar numbers between "+l+" and "+u);
      for(int i=l;i<=u;i++)
      {
         if(isKarprekar(i)) System.out.println("\t"+i);
      }
   }
}
```

```
class HammeringNo
{
```

```
  boolean isPrime(int n)
  {
    for(int i=2;i<=n/2;i++)
    {
      if(n%i==0) return false;
    }
    return true;
  }
  boolean isHammering(int n)
  {
    int p=2;
    int c=n;
    while(c>1)
    {
      if(!isPrime(p) || c%p!=0)
      {
        p++;
        continue;
      }
      c/=p;
      if(p>5) return false;
    }
    return true;
  }
}
```

```
//A unique-digit integer is a positive integer with no duplicate digits
import java.util.Scanner;
class Unique
{
  public static void main()
  {
    Scanner in=new Scanner(System.in);
    System.out.println("Enter:");
    System.out.println("1. To check if a number is unique ");
    System.out.println("2. To get a range of Unique numbers between any two numbers ");
    int i=in.nextInt();
    if(i==1)
    {
      System.out.println("Enter a number to check if it is unique ");
      int n=in.nextInt();
      if(isUnique(n)) System.out.println(n+" is Unique");
      else System.out.println(n+" is not Unique");
    }
    else if(i==2)
    {
      System.out.println("No. 1: ");
```

```
        int a=in.nextInt();
        System.out.println("No. 2: ");
        int b=in.nextInt();
        uniqueRange(a,b);
    }
    else
    {
        System.out.println("Invalid input:  Enter Again ");
        main();
        return;
    }
}
static int no_digits(int i)
{
    int n=0;
    while(i>0)
    {
        i/=10; n++;
    }
    return n;
}
static int[] extractDigits(int n, int l)
{
    int A[]=new int[l];int i=(l-1);
    while(n>0)
    {
        A[i]=n%10;
        n/=10; i--;
    }
    return A;
}
static boolean isUnique(int n)
{
    int l=no_digits(n);
    int D[]=new int[l];
    D=extractDigits(n,l);
    boolean unique=true;
    for(int i=0;i<=l-1;i++)
    {
        for(int j=0;j<=l-1;j++)
        {
            if(i==j) continue;
            if(D[i]==D[j]) unique=false;
        }
    }
    return unique;
}
static void uniqueRange(int m,int n)
{
    System.out.println("The Unique-Digit Integers Are:") ;
    int c=0;
    for(int i=m;i<=n;i++)
    {
        if(isUnique(i))
        {
```

```
            System.out.print(i+", ");
            c++;
         }
      }
      if(c==0) System.out.print("Nil ");
      System.out.println(" ");System.out.println(" ");
      System.out.println("Frequency of Unique-Digit Integers Is: "+c);
   }
}
```

```
//prg to find nth prime no. in a range
class PrimeAtPosition
{
   static boolean isPrime(int n) // to check if a prime
   {
      for(int i=2;i<(n/2+1);i++)
      {
         if(n%i==0) return false;
      }
      return true;
   }
   static void find_atPosition(int l,int u,int n)
   {
      int cp=0;
      for(int i=l;i<=u;i++)
      {
         if(isPrime(i)) cp++;
         if(cp==n)
         {
            System.out.println(n+"th prime no. found between "+l+" and "+u+" is "+i);
            break;
         }
      }
      if(cp<n)
      System.out.println("There is no"+n+"th prime no. between "+l+" and "+u);
   }
   public static void main(int L,int U,int n)
   {
      find_atPosition(L,U,n);
   }
}
```

```
import java.util.Scanner;
class Calculator
  {
   static double X, Y;
   static Scanner in=new Scanner(System.in);

   static double sqrt(int x)
   {
      double sq_rt=Math.sqrt(x);
      return sq_rt;
   }
   static double cbrt(int x)
   {
      double cb_rt=Math.cbrt(x);
      return cb_rt;
   }
   static double sqare(int x)
   {
      double sq=x*x;
      return sq;
   }
   static double cube(int x)
   {
      double cu=x*x*x;
      return cu;
   }
   static long nPr(int n, int r)
   {
      long a=1;
      for(int i=n;i>(n-r);i--)
      {
         a*=i;
      }
      return a;
   }
   static double expo(int x,int y)
   {
      double ex=Math.pow(x,y);
      return ex;
   }
   static double Addition(int x,int y)
   {
      double sum=x+y;
      return sum;
   }
   static double Subtraction(int x,int y)
   {
      double sub=x-y;
      return sub;
   }
   static double Multiplication(int x,int y)
   {
      double pro=x*y;
      return pro;
   }
```

```
static double DivisionQ(int x,int y)
{
   double Q=x/y;
   return Q;
}
static double DivisionR(int x,int y)
{
   double R=x%y;
   return R;
}
static double trig_fun(int Theta, int func)
{
   double out=0;
   switch(func)
   {
      case 1: out= Math.sin(Theta);
      break;
      case 2: out= Math.cos(Theta);
      break;
      case 3: out= Math.tan(Theta);
      break;
      case 4: out= 1/(Math.tan(Theta));
      break;
      case 5: out= 1/(Math.cos(Theta));
      break;
      case 6: out= 1/(Math.sin(Theta));
      break;
   }
   return out;
}
public static void main()
{
   int X=0;int Y=0; double result=0;
   for(;;)
   {  System.out.println((char)12);
      System.out.println("Choose any one of the following by entering the number adjacent to it");
      System.out.println("");
      System.out.println("1.  Square Root      2.  Cube Root");
      System.out.println("3.  Square          4.  Cube");
      System.out.println("5.  Addition        6.  Subtraction");
      System.out.println("7.  Multiplication  8.  Division");
      System.out.println("9.  Exponents       10. Trignometric Values");
      System.out.println("0. Exit");
      int a=in.nextInt();
      if(a==0) return;
      if(a==1||a==2||a==3||a==4)
      {  System.out.println("Please enter any one number")
         X=in.nextInt();
      }
      if(a==10)
      {
         System.out.println("Please enter any two numbers");
         X=in.nextInt();
         Y=in.nextInt();
      }
```

```
        if(a==5||a==6||a==7||a==8||a==9)
        {
           System.out.println("Please enter the no. for trignometric function to be used");
           System.out.println(" 1. sin    2. cos");
           System.out.println(" 3. tan    4. cot");
           System.out.println(" 5. sec    6. cosec");
           Y=in.nextInt();
           System.out.println("Enter the value of Theta");
           X=in.nextInt();
        }
        switch(a)
        {
           case 1:
           result=sqrt(X);
           break;

           case 2:
           result=cbrt(X);
           break;

           case 3:
           result=sqare(X);
           break;

           case 4:
           result=cube(X);
           break;

           case 5:
           result=Addition(X,Y);
           break;

           case 6:
           result=Subtraction(X,Y);
           break;

           case 7:
           result=Multiplication(X,Y);
           break;

           case 8:
           result=DivisionQ(X,Y);
           break;

           case 9:
           result=expo(X,Y);;
           break;

           case 10:
           result=trig_fun(Y,X);
           break;
        }
     }
  }
}
```

```
class FromDecimal
{
   public static int decimal_binary(int D)
   {
      int B=0;int p=0;
      while(D>0)
      {
         int d=D%2;
         B+=d*(Math.pow(10,p));
         D/=2; p++;
      }
      return B;
   }

   public static int decimal_octal(int D)
   {
      int O=0;int p=0;
      while(D>0)
      {
         int d=D%8;
         O+=d*(Math.pow(10,p));
         D/=8; p++;
      }
      return O;
   }

   public static String decimal_hexa(int D)
   {
      String H="" ; int p=0;
      char h=' ';
      while(D>0)
      {
         int d=D%16;
         if(d>9){ h= hexaDigits(d); H=h+H;}
         else
           H=d+H;
         D/=16;
      }
      return H;
   }

   public static char hexaDigits(int d)
   {
      char a=' ';
      switch(d)
      {
         case 1: case 2: case 3: case 4: case 5: case 6: case 7:case 8:case 9:
         a=(char)(d+48);
         break;

         case 10:
```

```
            a='A';
            break;

            case 11:
            a='B';
            break;

            case 12:
            a='C';
            break;

            case 13:
            a='D';
            break;

            case 14:
            a='E';
            break;

            case 15:
            a='F';
            break;
        }
        return a;
    }
}
```

```
class ToDecimal
{
    public static int binary_decimal(int B)
    {
        int D=0;int p=0;
        while(B>0)
        {
            int d=B%10;
            D+=d*(Math.pow(2,p));
            B/=10; p++;
        }
        return D;
    }
    public static int octal_decimal(int O)
    {
        int D=0;int p=0;
        while(O>0)
        {
            int d=O%10;
            D+=d*(Math.pow(8,p));
            O/=10; p++;
        }
```

```
    return D;
  }
  public static int hexa_decimal(String H)
  {
    int D=0,l=H.length();
    for(int p=0;p<l;p++)
    {
      char c=H.charAt(p);
      int d=hexaDigits(c);
      D+=d*(Math.pow(16,(l-1-p)));
    }
    return D;
  }
  private static int hexaDigits(char c)
  {
    int a=0;

    switch(c)
    {
      case '1':case '2':case '3':case '4':case '5':case '6':case '7':case '8':case '9':
      a=(int)c-48;
      break;

      case 'A':
      a=10;
      break;

      case 'B':
      a=11;
      break;

      case 'C':
      a=12;
      break;

      case 'D':
      a=13;
      break;

      case 'E':
      a=14;
      break;

      case 'F':
      a=15;
      break;
    }
    return a;
  }
}
```

```
class ToBinary
{
   public static int decimal_binary(int D)
   {
      int B=0;int p=0;
      while(D>0)
      {
         int d=D%2;
         B+=d*(Math.pow(10,p));
         D/=2; p++;
      }
      return B;
   }
   public static void octal_binary(int O)
   {
      int B=0;int p=0;
      while(O>0)
      {
         int d=O%10; int dB=decimal_binary(d);
         B+=dB*(Math.pow(1000,p));
         O/=10; p++;
      }
      System.out.println(B);
   }
   public static void hexa_binary(int H)
   {
      int B=0;int p=0;
      while(H>0)
      {
         int d=H%10; int dB=decimal_binary(d);
         B+=dB*(Math.pow(1000,p));
         H/=10; p++;
      }
      System.out.println(B);
   }
   private static int hexaDigits(char c)
   {
      int a=0;

      switch(c)
      {
         case '1':case '2':case '3':case '4':case '5':case '6':case '7':case '8':case '9':
         a=(int)c-48;
         break;

         case 'A':
         a=10;
         break;

         case 'B':
         a=11;
```

```
        break;

        case 'C':
        a=12;
        break;

        case 'D':
        a=13;
        break;

        case 'E':
        a=14;
        break;

        case 'F':
        a=15;
        break;
      }
      return a;
   }
}
```

```
class DecBin
{
   int D;
   static String d_b(int n)
   {
      if(n==1) return "1";
      return d_b(n/2)+n%2;
   }
}
```

```
class BinToDec
{
  int b_d(String B)
   {
      if(B.length()==0) return 0;
      else
       {
        int l=B.length()-1;
        int v=(B.charAt(0)-48)*(int)Math.pow(2,l);
        return v+b_d(B.substring(1));
      }
  }
}
```

```
class DecOct
{
   int D;
   static String d_o(int n)
   {
      if(n==0) return "";
      return d_o(n/8)+n%8;
   }
}
```

```
class DecHex
{
   int D;
   static String d_h(int n)
   {
      if(n==0) return "";
      String hD="";
      if(n/16<10) hD=""+n/16;
      else if(n/16==10) hD="A";
      else if(n/16==11) hD="B";
      else if(n/16==12) hD="C";
      else if(n/16==13) hD="D";
      else if(n/16==14) hD="E";
      else if(n/16==15) hD="F";
      return d_h(n/16)+hD;
   }
}
```

```
import java.util.Scanner;
class Date_DayNo
{
    int M[],DD,MM,YYYY,DN;
   Date_DayNo()
   {
      int a[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
      M=a;
   }
    void enterDate()
   {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter Date in");
      System.out.println("DD/MM/YYYY  format.");
      String D=in.next();
      String temp=D.substring(0,2);
      DD=Integer.parseInt(temp);

      temp=D.substring(3,5);
      MM=Integer.parseInt(temp);

      temp=D.substring(6);
      YYYY=Integer.parseInt(temp);
   }
   int convertDayNo()
   {

      for(int i=1;i<MM;i++)
      {
         DN+=M[i];
      }
      DN+=DD;
      return DN;
   }
   void isLeap()
   {
      if (YYYY%400==0 ||(YYYY%100!=0 && YYYY%4==0)) M[2]+=1;
   }
   public static void main()
   {
      Date_DayNo d=new Date_DayNo();
      d.enterDate();
      d.isLeap();
      d.convertDayNo();
   }
}

import java.util.Scanner;
class Days_Elapsed extends Date_DayNo
{
   int DN1,DN2,DN1_2;
```

```
    Days_Elapsed()
    {
       super();
       DN1=0;
       DN2=0;
       DN1_2=0;
    }
    void enter()
    {
       Date_DayNo d1=new Date_DayNo();
       d1.enterDate();
       d1.isLeap();
       DN1=d1.convertDayNo();

       Date_DayNo d2=new Date_DayNo();
       d2.enterDate();
       d2.isLeap();
       DN2=d2.convertDayNo();
    }
    void elapsed()
    {
       DN1_2=DN2-DN1;
    }
}




import java.util.Scanner;
class DayNo_Day
{
     int M[],DD,MM,YYYY,DN;
    DayNo_Day()
    {
       int a[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
       M=a;
       DD=0;
       MM=1;
    }
     void enterYearDayNo()
    {
       Scanner in=new Scanner(System.in);
       System.out.print("Enter Year in ");
       System.out.println("YYYY  format.");
       YYYY=in.nextInt();

       System.out.println("Enter Day no.");
       DN=in.nextInt();
    }
    void convertToDate()
    {
       int DaysLeft=DN;
       int i=1;
       while(DaysLeft>M[i])
```

```
      {
         DaysLeft-=M[i];
         i++;
         MM++;
      }
      DD=DaysLeft;
   }
   void isLeap()
   {
      if (YYYY%400==0 ||(YYYY%100!=0 && YYYY%4==0)) M[2]+=1;
   }
   void displayDate()
   {
      System.out.println("Day No. "+DN+" is:");
      System.out.println("    "+DD+"/"+MM+"/"+YYYY);
   }
   public static void main()
   {
      DayNo_Day d=new DayNo_Day();
      d.enterYearDayNo();
      d.isLeap();
      d.convertToDate();
      d.displayDate();
   }
}
```

```
/*This prg is based on a formula:
 * (Year Code + Month Code + Century Code + Date Number - Leap Year Code)% 7
 * which can be used to find the day on any date.
 *
 * How to Calculate the Day of the Week from Any Date(Procedure)-
 *  <https://artofmemory.com/blog/how-to-calculate-the-day-of-the-week/> (biblography)
*/
import java.util.Scanner;
public class AnyCalender
{
   int YearCode,CenturyCode,MonthCode,DateNumber,LeapYearCode,Day_No,Month;
   boolean LeapYear,J_F;
   String Day;
   private void Year_Cent(int yr)
   {
      int Cent=yr/100;
      if(Cent==17||Cent==21||Cent==25||Cent==29||Cent==33)   CenturyCode=4;
      if(Cent==18||Cent==22||Cent==26||Cent==30||Cent==34)   CenturyCode=2;
      if(Cent==19||Cent==23||Cent==27||Cent==31||Cent==35)   CenturyCode=0;
      if(Cent==20||Cent==24||Cent==28||Cent==32||Cent==36)   CenturyCode=6;
```

```
   int YY=yr%(Cent*100);
   YearCode=(YY+(YY/4))%7;
   if(yr%4!=0) LeapYear=false;
   else if(yr%4==0)
   {
      if(yr%100!=0) LeapYear=true;
      else if(yr%100==0)
      {
         if(yr%400==0) LeapYear=true;
         if(yr%400!=0) LeapYear=false;
      }
   }
   if(LeapYear==true)    LeapYearCode=1;
   if(LeapYear==false)   LeapYearCode=0;
}
private void MONTH(int month)
{
   if (month>12) System.out.println("Invalid Month");
   Month=month;
   if(month==1||month==10) MonthCode=0;
   if(month==2||month==3||month==11) MonthCode=3;
   if(month==4||month==7) MonthCode=6;
   if(month==5) MonthCode=1;
   if(month==6) MonthCode=4;
   if(month==8) MonthCode=2;
   if(month==9||month==12) MonthCode=5;
   if(month==1||month==2)  J_F=true;
}
private void Calculate()
{
   DateNumber=1;
   if(LeapYearCode==1 && J_F==true)  LeapYearCode=1;
   else                          LeapYearCode=0;

   Day_No=(YearCode+MonthCode+CenturyCode+DateNumber-LeapYearCode)%7;
}
private void SMTWTFS()
{
   System.out.println("");
   System.out.print("\tSUN\t");
   System.out.print("MON\t");
   System.out.print("TUES\t");
   System.out.print("WED\t");
   System.out.print("THUS\t");
   System.out.print("FRI\t");
   System.out.print("SAT\t");
   System.out.println("");
}
private void whichDay1()
{
   for(int i=0;i<Day_No;i++)
   {
      System.out.print("\t   ");
      System.out.print("\t 1 ");
   }
```

```
    }
    private void printingDays_n_dates()
    {
        int date=DateNumber+1;
        int day=Day_No+1;
        int Mdays=0;
        if(Month%2==0)
        {
            if(LeapYearCode==1 && Month/2==1) Mdays=29;
            else if(LeapYearCode==0 && Month/2==1) Mdays=28;
        }
        if(Month==1||Month==3||Month==5||Month==7||Month==8||Month==10||Month==12) Mdays=31;
        if(Month==4||Month==6||Month==9||Month==11) Mdays=30;

        for(int i=1;i<Mdays;i++)
        {
            if(day>6)
            {
                day%=7;
                System.out.println("");
            }
            System.out.print("\t "+date+" ");
            date++;
            day++;
        }
    }
    public void main()
    {
        System.out.println((char)12);
        Scanner input=new Scanner(System.in);
        System.out.println("Enter the year in YYYY format");
        int year=input.nextInt();
        Year_Cent(year);

        System.out.println("Enter Month no.");
        int Month=input.nextInt();
        MONTH(Month);

        Calculate();
        SMTWTFS();
        whichDay1();
        printingDays_n_dates();
    }
} //THIS PRG WAS MADE IN THE DECEMBER OF 2020
```

ARRAYS

```
class Linear_Search
{
   public static void search(int s,int list[])
   {
      for(int i=0;i<list.length;i++)
      {
         if(s==list[i])
         {
            System.out.println(s+" is found at position "+(i+1));
            return;
         }
      }
      System.out.println(s+" not found.");
   }
}
```

```
class BinarySearch
{
   public static void main(int s,int list[])
   {
      int first=0;
      int last=list.length-1;

      while(first<last)
      {
         int mid=(first+last)/2;
         if(list[mid]==s)
         {
            System.out.println(s+" is found at "+(mid+1));
            return;
         }
         else if(list[mid]>s)
           last=mid-1;
         else
           first=mid+1;
      }
      System.out.println(s+" not found.");
   }
}
```

```
public class selection_sort//ascending
```

```
{
   public static void main(int l[])
   {
      for(int i=0;i<=l.length-1;i++)
      {
         for(int j=i+1;j<=l.length-1;j++)
         {
            if(l[i]>l[j])
            {
               int c=l[i];
               l[i]=l[j];
               l[j]=c;
            }
         }
      }
      for(int i=0;i<=l.length-1;i++)
      {
         System.out.println(l[i]+"");
      }
   }
}



class BubbleSort
{
   int A[];
   BubbleSort(int a[])
   {
      A=a;
   }
   void sort()
   {
      for(int i=0;i<A.length-1;i++)
      {
         for(int j=0;j<A.length-1-i;j++)
         {
            if(A[j]>A[j+1])
            {
               int c=A[j];
               A[j]=A[j+1];
               A[j+1]=c;
            }
         }
      }
   }
   void display()
   {
      for(int i=0;i<A.length;i++)
      {
         System.out.println(A[i]+", ");
      }
   }
}
class Insertion
```

```
{
   int A[];int c;
   Insertion(int in[])
   {
      A=new int[(in.length)+1];
      for(int i=0;i<in.length;i++)
      {
         A[i]=in[i];
      }
      c=A.length-1;
      A[c]=0;
   }
   public void Insert(int in,int p)
   {
      for(int i=c;i>p-1;i--)
      {
         A[i]=A[i-1];
      }
      A[p-1]=in;
   }
   void display()
   {
      for(int i=0;i<=c;i++)
      {
           System.out.println(A[i]);
      }
   }
   public static void main(int x[],int in,int p)
   {
      Insertion X=new Insertion(x);
      X.Insert(in,p);
      X.display();
   }
}
```

```
class InsertionSort
  {
   int N[];

   void input(int n[])
       { N=n; }

  void arrange()
   {
   for(int i=1;i<N.length;i++)
    {
     int T=N[i];
    int j=i-1;
    while(j>=0 && T<N[j])
      {
      N[j+1]=N[j];
      j--;
```

```
    }
   N[j+1]=T;
   }
 }

void display()
 {System.out.println("the List is as Follows");
  for(int i=0;i<N.length;i++)
   {
    System.out.println(N[i]);
   }
 }
```

```
class Deletion
{
    int A[]; int c;
    Deletion(int A[])
    {
        this.A=A;
        c=A.length;
    }
    public void Delete(int p)
    {
        for(int i=p-1;i<A.length-1;i++)
        {
           A[i]=A[i+1];
        }
        A[A.length-1]=0;c--;
    }
    void display()
    {
        for(int i=0;i<c;i++)
        {
             System.out.println(A[i]);
        }
    }
    public static void main(int x[],int p)
    {
        Deletion X=new Deletion(x);
        X.Delete(p);
        X.display();
    }
}
class Merging //prints duplicates too
{
    int R[],A[],B[]; int rL;
    Merging(int A[],int B[])
    {
        this.A=A;
```

```
    this.B=B;
    rL=A.length+B.length;
    this.R=new int[rL];
  }
  public void Merge()
  {
    int i=0,j=0,k=0;
    while(i<A.length||j<B.length)
    {
      if(i==A.length && j<B.length)
      {
        R[k]=B[j];
        j++;  k++;
      }
      else if(j==B.length && i<A.length)
      {
        R[k]=A[i];
        i++;  k++;
      }
      else if(A[i]<B[j])
      {
        R[k]=A[i];
        i++;  k++;
      }
      else if(B[j]<A[i])
      {
        R[k]=B[j];
        j++;  k++;
      }
      else if(B[j]==A[i])
      {
        R[k]=B[j];
        j++;  k++;
        R[k]=A[i];
        i++;  k++;
      }
    }
  }
  void display()
  {
    for(int i=0;i<rL;i++)
    {
        System.out.println(R[i]);
    }
  }
  public static void main(int x[],int y[])
  {
    Merging X=new Merging(x,y);
    X.Merge();
    X.display();
  }
}
```

```
class MatrixFunctionsChecks
{
   int[][] transpose(int M[][])
   {
      int N[][]=new int[M.length][M[0].length];
      for(int i=0;i<M.length;i++)
      {
         for(int j=0;j<M[0].length;j++)
         {
            M[i][j]=N[j][i];
         }
      }
      return N;
   }
   boolean isComparable(int A[][],int B[][])
   {
      if(A.length==B.length && A[0].length==B[0].length)
      return true;
      return false;
   }
   boolean isEqual(int A[][],int B[][])
   {
      if(isComparable(A,B)==false) return false;

      for(int i=0;i<A.length;i++)
```

```
        {
            for(int j=0;j<A.length;j++)
            {
                if(A[i][j]!=B[i][j]) return false;
            }
        }
        return true;
    }
    int[][] negative(int A[][])
    {
        int N[][]=new int[A.length][A[0].length];
        for(int i=0;i<A.length;i++)
        {
            for(int j=0;j<A[0].length;j++)
            {
                N[i][j]=-A[i][j];
            }
        }
        return N;
    }
    boolean isSymmetric(int A[][])
    {
        int B[][]=transpose(A);
        if(isEqual(A,B)) return true;
        return false;
    }
    boolean isSkewSymmetric(int A[][])
    {
        int B[][]=negative(transpose(A));
        if(isEqual(A,B)) return true;
        return false;
    }
    int[][] multiplyScalar(int k,int A[][])
    {
        for(int i=0;i<A.length;i++)
        {
            for(int j=0;j<A[0].length;j++)
            {
                A[i][j]*=k;
            }
        }
        return A;
    }
}
```

```
import java.util.Scanner;
class SortRow
{
```

```
    int A[][];
    int m,n;
    SortRow(int M,int N)
    {
       m=M;
       n=N;
       A=new int[m][n];
    }
    void inputMatrix()
    {
       Scanner in=new Scanner(System.in);
       for(int i=0;i<m;i++)
       {
          for(int j=0;j<n;j++)
          {
             A[i][j]=in.nextInt();
          }
       }
    }
    int[] sort(int a[])
    {
       for(int i=0;i<a.length-1;i++)
       {
          for(int j=i+1;j<a.length;j++)
          {
             if(a[i]>a[j])
             {
                int c=a[i];
                a[i]=a[j];
                a[j]=c;
             }
          }
       }
       return a;
    }
    void sortRows()
    {
       for(int i=0;i<n;i++)
       {
          A[i]=sort(A[i]);
       }
    }
    void display()
    {
       for(int i=0;i<m;i++)
       {
          for(int j=0;j<n;j++)
          {
             System.out.print(" "+A[i][j]+" ");
          }
          System.out.println();
       }
    }
}
```

```
class SeriesSpiral
{
   int S,SA[][],r,c;
   SeriesSpiral(int s)
   {
      S=s;
      SA=new int[s][s];
      r=s;c=s;
   }
   public void createSpiral()
   {
      int k=1;
      for(int i=0;i<r-1;i++)
      {
         for(int j=0;j<c-1;j++)
         {
            SA[i][j]=k;
            if(j==(c-1)) {j--;i++;}
         }
      }
   }
}
```

```
import java.util.Scanner;
class Q2_09
{
   int M[][],m,n;
   Q2_09(int M,int N)
   {
      m=M;
      n=N;
      this.M=new int[M][N];
   }
   void inputMatrix()
   {
      Scanner in=new Scanner(System.in);
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<n;j++)
         {
            M[i][j]=in.nextInt();
         }
      }
   }
   void display()
   {
      for(int i=0;i<m;i++)
      {
```

```
      for(int j=0;j<n;j++)
      {
         System.out.print(" "+M[i][j]+" ");
      }
      System.out.println();
   }
}
void displayBoundary()
{
   for(int i=0;i<m;i++)
   {
      for(int j=0;j<n;j++)
      {
         if(i==0||j==0||i==(m-1)||j==(n-1))System.out.print(" "+M[i][j]+" ");
         else System.out.print("   ");
      }
      System.out.println();
   }
}
int sumBoundary()
{
   int S=0;
   for(int i=0;i<m;i++)
   {
      for(int j=0;j<n;j++)
      {
         if(i==0||j==0||i==(m-1)||j==(n-1))S+=M[i][j];
      }
      System.out.println();
   }
   return S;
}
void rearrangeBoundary()
{
   Scanner in=new Scanner(System.in);
   int B[]=new int[m*n];
   int k=0;
   for(int i=0;i<m;i++)
   {
      for(int j=0;j<n;j++)
      {
         if(i==0||j==0||i==(m-1)||j==(n-1))
         {
            B[k]=M[i][j];
            k++;
         }
      }
   }
   sort(B);
   int l=k;
   k=0;


   for(int i=0,j=0;j<m;j++)
   {
```

```
        M[i][j]=B[k];
        k++;
    }
    for(int i=1,j=(n-1);i<m;i++)
    {
        M[i][j]=B[k];
        k++;
    }
    for(int i=(m-1),j=(n-1)-1;j>=0;j--)
    {
        M[i][j]=B[k];
        k++;
    }
    for(int i=(m-1)-1,j=0;i>0;i--)
    {
        M[i][j]=B[k];
        k++;
    }
}
void sort(int A[])
{
    for(int i=0;i<A.length-1;i++)
    {
        for(int j=i+1;j<A.length-1;j++)
        {
            if(A[i]>A[j])
            {
                int c=A[i];
                A[i]=A[j];
                A[j]=c;
            }
        }
    }
}
public static void main()
{
    Scanner in=new Scanner(System.in);
    System.out.print("M=");
    int M=in.nextInt();
    System.out.print("N=");
    int N=in.nextInt();
    System.out.println(" ");

    Q2_09 mm=new Q2_09(M,N);
    mm.inputMatrix();

    System.out.println("ORIGINAL MATRIX ");
    mm.display();
    mm.rearrangeBoundary();
    System.out.println("REARRANGED MATRIX ");
    mm.display();
    System.out.println("BOUNDARY ELEMENTS ");
    mm.displayBoundary();

    System.out.println(" ");
```

```
        System.out.println("SUM OF OUTER ROW AND COLUMN ELEMENTS = "+mm.sumBoundary());
    }
}
```

```
class CarSales
{
    int N[][]=new int[6][5];
    int D[]=new int[6];
    int M[]=new int[5];
    String  day[]={"Mon","Tues","Wed","Thus","Fri","Sat"};
    String  model[]={"Alto","Swift","Desire","Wagon-R","Baleno"};
    CarSales(int n[][])
    {
        N=n;
    }
    public void sortSale()
    {
        for(int i=0;i<6;i++)
        {
            for(int j=0;j<5;j++)
            {
                D[i]+=N[i][j];
            }
        }
        for(int j=0;j<5;j++)
        {
            for(int i=0;i<6;i++)
            {
                M[j]+=N[i][j];
            }
        }
    }
    public void saleDay()
    {
        for(int i=0;i<6;i++)
        {
            System.out.println(D[i]+" ");
        }
    }
    public void saleModelWeekly()
    {
        for(int i=0;i<5;i++)
        {
            System.out.println(M[i]+" ");
        }
    }
}
```

```
import java.util.Scanner;
class Sort2D
{
   int A[][],B[],C[][],r,c;
   void input()
   {
      Scanner S=new Scanner(System.in);
      System.out.println("Enter no.of rows ");
      r=S.nextInt();
      System.out.println("Enter no.of columns ");
      c=S.nextInt();
      A=new int[r][c];C=new int[r][c];
      System.out.println("Enter the elements of Matrix ");
      for(int i=0;i<r;i++)
      {
         System.out.println("Row "+(i+1));
         for(int j=0;j<c;j++)
         {
            System.out.println((j+1)+" :");
            A[i][j]=S.nextInt();
         }
         System.out.println(" ");
      }
   }
   void convertAtoB()
   {
      B=new int[r*c];int k=0;
      for(int i=0;i<r;i++)
      {
         for(int j=0;j<c;j++)
         {
            B[k]=A[i][j]; k++;
         }
      }
   }
   void sort()
   {
      for(int i=0;i<B.length-1;i++)
      {
         for(int j=0;j<B.length-1-i;j++)
         {
            if(B[j]>B[j+1])
            {
               int b=B[j];
               B[j]=B[j+1];
               B[j+1]=b;
```

```
            }
        }
    }
  }
  void convertBtoC()
  {
    int k=0;
    for(int i=0;i<r;i++)
    {
      for(int j=0;j<c;j++)
      {
        C[i][j]=B[k]; k++;
      }
    }
  }
  void display_sorted()
  {
    for(int i=0;i<r;i++)
    {
      for(int j=0;j<c;j++)
      {
        System.out.print(C[i][j]+"  ");
      }
      System.out.println("  ");
    }
  }
  public static void main()
  {
    Sort2D A=new Sort2D();
    A.input();
    A.convertAtoB();
    A.sort();
    A.convertBtoC();
    A.display_sorted();
  }
}
```

```
class MagicSquareOdd
{
  static boolean isMultiple(int n,int m)
  {
    if(n%m==0)return true;
    return false;
  }
  public static void main(int rc)
  {
    int A[][]=new int[rc][rc];
    int i=0,j=0;
```

```
      for(int n=1;n<=rc*rc;n++)
      {
         if(n==1)j=(rc-1)/2;
         else if(isMultiple(n-1,rc))i++;
         else{--i;++j;}
         //System.out.print("");
         if(i<0)i=rc-1;

         if(j==rc)j=0;
         A[i][j]=n;
      }
      System.out.println("");
      for(i=0;i<rc;i++)
      {
         for(j=0;j<rc;j++)
         {
            System.out.print(A[i][j]+"  ");
         }
         System.out.println("");
      }
      System.out.println("");
      DisplayOrganised.diplay(A);
   }
}
```

```
import java.util.Scanner;
class Markov
{
   boolean Mark;
   double M[][];
   int N;
   Markov ()
   {
      Mark=true;
   }
   void input()
   {
      Scanner in=new Scanner(System.in);
      System.out.print("Enter the size for the matrix:");
      N=in.nextInt();
      if(N<3 || N>9)
      {
         System.out.print("Invalid input");
         input();
         return;
      }
      M=new double[N][N];
      System.out.println("");
      System.out.println("Enter Matrix elements:");
      for(int i=0;i<N;i++)
```

```
      {
         for(int j=0;j<N;j++)
         {
            M[i][j]=in.nextDouble();
            if(M[i][j]<0)
            {
               System.out.print("Invalid input, Enter again");
               j--;
            }
         }
      }
   }

   void checkMarkov()
   {
      for(int i=0;i<N;i++)
      {
         double S1=0;double S2=0;
         for(int j=0;j<N;j++)
         {
            S1+=M[i][j];

            S2+=M[j][i];
         }
         if(S1!=1) Mark=false;
         if(S2!=1) Mark=false;
         System.out.println("S1="+S1+", S2="+S2);
      }
   }
   public static void main()
   {
      Markov m=new Markov();
      m.input();
      m.checkMarkov();
      if(m.Mark) System.out.println("Yes, it is a Markov Matrix");
      else System.out.println("No, it is not a Markov Matrix");
   }
}




class Sum
{
   static int A[][];
   static void sumArray()
   {
      int sum=0;
      for(int i=0;i<A.length;i++)
      {
         for(int j=0;j<A[0].length;j++)
         {
            sum+=A[i][j];
```

```
        }
      }
      System.out.println(sum);
   }
   static void sumRow()
   {
      for(int i=0;i<A.length;i++)
      {
         int sum=0;
         for(int j=0;j<A[0].length;j++)
         {
            sum+=A[i][j];
         }
         System.out.println(sum);
      }
   }
   static void sumColumn()
   {
      for(int i=0;i<A.length;i++)
      {
         int sum=0;
         for(int j=0;j<A[0].length;j++)
         {
            sum+=A[j][i];
         }
         System.out.print(sum+"  ");
      }
   }
   static void sumDiagonal()
   {
      int s1=0,s2=0;
      for(int i=0;i<A.length;i++)
      {
         s1+=A[i][i];
         s1+=A[i][A.length-1-i];
      }
      System.out.print(s1+" & "+s2);
   }
}
```

```
class Product
{
   static int A[][];
   static void proRow()
   {
      for(int i=0;i<A.length;i++)
      {
         long pro=1l;
```

```
        for(int j=0;j<A[0].length;j++)
        {
           pro*=A[i][j];
        }
        System.out.println(pro);
     }
  }
  static void proColumn()
  {
     for(int i=0;i<A.length;i++)
     {
        long pro=1l;
        for(int j=0;j<A[0].length;j++)
        {
           pro*=A[j][i];
        }
        System.out.print(pro+"  ");
     }
  }
  static void proDiagonal()
  {
     int p1=1,p2=1;
     for(int i=0;i<A.length;i++)
     {
        p1*=A[i][i];
        p1*=A[i][A.length-1-i];
     }
     System.out.print(p1+" & "+p2);
  }
}
```

STRING

```
import java.util.*;
class LongestWord
{
   String S="";
   String W[]=new String[25];
   int nw;
   public LongestWord(String Sentence)
   {
      S=Sentence;
      for(int i=0;i<W.length;i++)
      {
         W[i]="";
      }
   }
   void extractWords()
   {
      int L=S.length();
      int k=0;
      for(int i=0;i<L;i++)
      {
         char x=S.charAt(i);
         if(x==' ' || x=='.'||x==','|| x=='?'||x=='!'|| x==';'||x==':')
         {
            k++;
            nw++;
         }
         else W[k]+=x;
      }
   }
   void longest()
   {
      String longest=W[0];
      int longL=W[0].length();
      for(int i=1;i<nw;i++)
      {
         int l=W[i].length();
         if(longL<l)
         {
            longL=l;
            longest=W[i];
         }
      }
      System.out.println(longest);
   }
   public static void main()
   {
      System.out.println("Enter a Sentence");
      Scanner in=new Scanner(System.in);
      String Sentence=in.nextLine();
      LongestWord L=new LongestWord(Sentence);
      L.extractWords();
      L.longest();
   }
}
```

```
class remove_vowels
{
   String S;
   String newS;
   void input(String in)
   {
      S=in;
      newS="";
   }
   void remove()
   {
      for(int i=0;i<S.length();i++)
      {
         char c=S.charAt(i);
         if(c=='a' || c=='e' ||c=='i' || c=='o' ||c=='u' ||c=='A' || c=='E' ||c=='I' || c=='O' ||c=='U')
         continue;
         newS+=c;
      }
   }
   void display()
   {
      System.out.println(newS);
   }
   public static void main(String in)
   {
      remove_vowels n1=new remove_vowels();
      n1.input(in);
      n1.remove();
      n1.display();
   }
}
```

```
public class Words_vowelStartEnd
{
   String S;
   String W[]=new String[30];
   int wc=0;

  Words_vowelStartEnd(String S)
   {
      this.S=S;
      for(int i=0;i<30;i++)
      {
         W[i]="";
      }
   }

   void extract()
   {
      int k=0;int l=S.length();
      for(int i=0;i<l;i++)
```

```
    {
      char x=S.charAt(i);
      if(x==' '||x=='.')k++;
      else W[k]+=x;
    }
    wc=k;
  }
  void vowel2()
  {
    for(int i=0;i<=wc;i++)
    {
      boolean vowelF_L=false;
      int c=0;int l=W[i].length();
      char x=W[i].charAt(0);char y=W[i].charAt(l-1);
      if(x=='a'||x=='e'||x=='i'||x=='o'||x=='u') vowelF_L=true;
      if(y=='a'||y=='e'||y=='i'||y=='o'||y=='u') vowelF_L=true;
      if(vowelF_L) System.out.println(W[i]);
    }
  }
  public static void main(String args[])
  {
    Words_vowelStartEnd ev=new Words_vowelStartEnd(args[0]);
    ev.extract();
    ev.vowel2();
  }
}




public class ExtractWordsA
{
  String S="";
  String W[]=new String[25];
  int nw;
  public ExtractWordsA(String Sentence)
  {
    S=Sentence;
    for(int i=0;i<W.length;i++)
    {
      W[i]="";
    }
  }
  void extractWords()
  {
    int L=S.length();
    int k=0;
    for(int i=0;i<L;i++)
    {
      char x=S.charAt(i);
      if(x==' ' || x=='.'||x==','|| x=='?'||x=='!'|| x==';'||x==':')
      {k++;nw++;}
      else W[k]+=x;
    }
```

```
    }
    public static String[] extract_Words(String S)
    {
        int L=S.length();
        String W[]=new String[25];
        int nw=0;
        W[0]="";
        int k=0;
        for(int i=0;i<L;i++)
        {
            char x=S.charAt(i);
            if(x==' ' || x=='.'||x==','|| x=='?'||x=='!'|| x==';'||x==':')
            {k++;nw++;W[k]="";}
            else W[k]+=x;
        }
        return W;
    }
}
```

```
class Extract_Vowels2
 {
    String S;
    String W[]=new String[30];
    int wc=0;

    Extract_Vowels2(String S)
    {
        this.S=S;
        for(int i=0;i<30;i++)
        {
            W[i]="";
        }
    }

    void extract()
    {
        int k=0;int l=S.length();
        for(int i=0;i<l;i++)
        {
            char x=S.charAt(i);
            if(x==' '||x=='.')k++;
            else W[k]+=x;
        }
        wc=k;
    }


    void vowel2()
    {
        for(int i=0;i<=wc;i++)
```

```
    {
      int c=0;
      for(int j=0;j<W[i].length();j++)
      {
        char x=W[i].charAt(j);
        if(x=='a'||x=='e'||x=='i'||x=='o'||x=='u') c++;
      }
      if(c>=2) System.out.println(W[i]);
    }
  }

  public static void main(String args[])
  {
    Extract_Vowels2 ev=new Extract_Vowels2(args[0]);
    ev.extract();
    ev.vowel2();
  }
}




class WeightWords
{
  String W[],S;
  int V[],nw;
  WeightWords(String s)
  {
    S=s.trim();
    S=S.toUpperCase();
    W=new String[10];
    V=new int[10];
    for(int i=0;i<10;i++)
     W[i]="";
  }
  int weigh(String w)
  {
    if(w.length()==1) return (int)w.charAt(0)-64;
    return (int)w.charAt(0)-64+weigh(w.substring(1));
  }
  void extract_weigh()
  {
    int x=0;
    for(int i=0;i<S.length();i++)
    {
      if(S.charAt(i)==' ')
      {
        x++;nw++; continue;
      }
      W[x]+=S.charAt(i); V[x]+=weigh(S.substring(i,i+1));
    }
    nw++;
  }
  void arrange()
  {
```

```
      for(int i=0;i<nw;i++)
      {
         for(int j=i+1;j<nw;j++)
         {
            if(V[i]>V[j])
            {
               int v=V[i];  String w=W[i];
               V[i]=V[j];   W[i]=W[j];
               V[j]=v;      W[j]=w;
            }
         }
      }
   }
   void display()
   {
      for(int i=0;i<nw;i++)
      {
         System.out.print(W[i]+" ");
      }
      System.out.println("  ");
      for(int i=0;i<nw;i++)
      {
         System.out.print(" "+V[i]+" ");
      }
   }
   public static void main(String args[])
   {
      WeightWords WW=new WeightWords(args[0]);
      WW.extract_weigh();
      System.out.println("Original Sentence: ");
      WW.display();
      WW.arrange();
      System.out.println("  ");
      System.out.println("Arranged Sentence: ");
      WW.display();
   }
}
```

```
//isc practical 2003
class Decrypt2k3_1
 {
    String CT;
    byte shift;

    Decrypt2k3_1(String CT, byte shift)
     {
        this.CT=CT;
        if(shift<2 || shift>25)
                    System.out.println("invalid shift value");
       shift--;
```

```
        this.shift=shift;

    }

   void decryptNow()
    {
       int l=CT.length();
       String DT="";
       for(int i=0;i<l;i++)
        {
           int x=CT.charAt(i);
           if(x==32) continue;
           x=x+shift;
           if(x>90) x-=26;
           DT=DT+(char)x;
        }
       System.out.println("Decrypted text is ");
       for(int i=0;i<DT.length();i++)
        {
           if (DT.charAt(i)=='Q' && DT.charAt(i+1)=='Q')
                                    {System.out.print(" ");
                                     i++;}
           else System.out.print(DT.charAt(i));
        }
      }
   }
```

```
class SortSentenceWords
{
   String Sentence,Words[];
   int wL[],nW;
   SortSentenceWords(String S)
   {
      Sentence=S;
      Words=new String[10];
      for(int i=0;i<10;i++)
      {
         Words[i]=" ";
      }
   }
   void extract()
   {
      Sentence.trim();
      int k=0;
      for(int i=0;i<Sentence.length();i++)
      {
         char x=Sentence.charAt(i);
         if(x=='.') break;
         if(x==' ') k++;
```

```
        else Words[k]+=x;
      }
      nW=k+1;
   }
   void sortWords()
   {
      wL=new int[nW];
      for(int i=0;i<nW;i++)
      {
         wL[i]=Words[i].length();
      }

      for(int i=0;i<nW-1;i++)
      {
         for(int j=0;j<nW-i-1;j++)
         {
            if(wL[j]>wL[j+1])
            {
               int c=wL[j];   String cW=Words[j];
               wL[j]=wL[j+1]; Words[j]=Words[j+1];
               wL[j+1]=c;     Words[j+1]=cW;
            }
         }
      }
   }
}
```

```
import java.util.Scanner;
class Q2_11
{
   String S[];
   Q2_11(int n)
   {
      S=new String[n];
   }
   void input()
   {
      Scanner in=new Scanner(System.in);
      for(int i=0;i<S.length;i++)
      {
         S[i]=in.nextLine();
      }
   }
   void printEncoded(String s)
   {
      String n="";
      for(int i=0;i<s.length();i++)
      {
         if(s.charAt(i)=='.' || s.charAt(i)==' ')
```

```
        {
            n+=s.charAt(i);
            continue;
        }
        int p=s.charAt(i);
        p+=2;
        if(p>90) p-=26;
        n+=(char)p;
      }
      System.out.println(n);
   }
   void print()
   {
      for(int i=0;i<S.length;i++)
      {
         if(i%2==0) printEncoded(S[i]);
         else System.out.println(S[i]);
      }
   }
   public static void main()
   {
      Scanner in=new Scanner(System.in);
      System.out.print("n=");
      int n=in.nextInt();
      if(n<1||n>10)
      {
         System.out.print("INVALID ENTRY");
         return;
      }
      Q2_11 s=new Q2_11(n);
      System.out.println(" ");
      s.input();
      System.out.println(" ");
      s.print();
   }
}
```

```
class NumbersToWords
{
   int n;
   String w;
   NumbersToWords (int N)
   {
      n=N;
      w="";
   }
   private String toWords(int N)
   {
      if(N==1) return "ONE";
```

```
    else if(N==2) return "TWO";
    else if(N==3) return "THREE";
    else if(N==4) return "FOUR";
    else if(N==5) return "FIVE";
    else if(N==6) return "SIX";
    else if(N==7) return "SEVEN";
    else if(N==8) return "EIGHT";
    else if(N==9) return "NINE";
    else if(N==10) return "TEN";
    else if(N==11) return "ELEVEN";
    else if(N==12) return "TWELVE";
    else if(N==13) return "THIRTEEN";
    else if(N==14) return "FOURTEEN";
    else if(N==15) return "FIFTEEN";
    else if(N==16) return "SIXTEEN";
    else if(N==17) return "SEVENTEEN";
    else if(N==18) return "EIGTEEN";
    else if(N==19) return "NINETEEN";
    else if(N>19 && N<30) return "TWENTY "+toWords(N%10);
    else if(N>29 && N<40) return "THIRTY "+toWords(N%10);
    else if(N>39 && N<50) return "FOURTY "+toWords(N%10);
    else if(N>49 && N<60) return "FIFTY "+toWords(N%10);
    else if(N>59 && N<70) return "SIXTY "+toWords(N%10);
    else if(N>69 && N<80) return "SEVENTY "+toWords(N%10);
    else if(N>79 && N<90) return "EIGHTY "+toWords(N%10);
    else if(N>89 && N<100) return "NINTY "+toWords(N%10);
    else return "";
  }
  void numToWords()
  {
    w=toWords(n/100);
    if(n>=100)
    {
      w+=" HUNDRED ";
      if(n%100>0)w+="AND ";
    }
    w+=toWords(n%100);
  }
}
```

RECURSION

```
class Factorial
{
   static int F=1;
   private static int fac(int N)
   {
      if(N==1) return 1;
       F=N*fac(N-1);
           return F;
   }
   private static void display()
   {
      System.out.println(F);
   }
   public static void main(int n)
   {
      System.out.print("Factorial of "+n+"= ");
      fac(n);
      display();
   }
}
```

```
class Power
{
   static int P=1;
   static int pow(int n,int i)
   {
      if(i==0 )return 1;
      P=n*pow(n,i-1);
      return P;
   }
   public static void main(int n,int p)
   {
      System.out.print(n+"^"+p+"="+pow(n,p));
   }
}
```

```
class Multiple3
{
   public static int getSum(int n)
   {
      if(n==1) return 3;
      return n*3+getSum(n-1);
   }

}

class Perfect
```

```
{
   int num;
   Perfect(int nn)
   {
      num=nn;
   }

   int sumFac(int f)
   {
      if(f==1) return 1;
      if( num%f==0) return f+sumFac(f-1);
      return sumFac(f-1);
   }

   boolean isPerfect()
   {
      if(sumFac((num/2)+1)==num) return true;
      return false;
   }

   public static void main(int n)
   {
      Perfect N=new Perfect(n);
      if(N.isPerfect()) System.out.println(n+" is a Perfect No.");
      else System.out.println(n+" is not a Perfect No.");
   }
}




class Reverse
{
   static String reverseL(String nm)
   {
      if(nm.length()==1) return nm;
      return reverseL(nm.substring(1))+nm.charAt(0);
   }

   static String reverseR(String nm)
   {
      if(nm.length()==1) return nm;
      return nm.charAt(nm.length()-1)+reverseR(nm.substring(0,nm.length()));
   }
}









class DiceriumR
{
```

```
    int num,nD;
    DiceriumR(int nn)
    {
        num=nn;
    }
    int no_digits(int n)
    {
        int c=0,copy=n;
        while(copy>0)
        {
            copy/=10;
            c++;
        }
        nD=c;
        return c;
    }
    int powersN(int n,int p)
    {
        if(n<10) return n;
        return (int)Math.pow(n%10,p)+powersN(n/10,p-1);
    }
    int powersN(int n)
    {
        if(n<10) return n;
        int o=n%10;
        return (int)Math.pow(o,no_digits(n))+powersN((int)(n-o)/10);
    }
    boolean isDicerium()
    {
        if(num==powersN(num)) return true;
        return false;
    }
    public static void main(int n)
    {
        DiceriumR D=new DiceriumR(n);
        if(D.isDicerium()) System.out.println(n+" is a Dicerium no.");
        else System.out.println(n+" is not a Dicerium no.");
    }
}
```

INHERITANCE

```
class Rectangle
{
   double L,B;
   Rectangle(double l,double b)
   {
      L=l;
      B=b;
   }
   void display()
   {
      System.out.println("Length= "+L);
      System.out.println("Breath= "+B);
   }
}
class Perimeter extends Rectangle
{
   double P;
   Perimeter(double x,double y)
   {
      super(x,y);
      P=2*(L+B);
   }
   void display()
   {
      super.display();
      System.out.println("Perimeter= "+P);
   }
}
```

```
class Stat
{
   int X[],F[],CF[],sF,l;
   Stat(int x[],int f[])
   {
      X=x;
```

```
        F=f;
        l=F.length;
        CF=new int[l];
        CF[0]=F[0];
        for(int i=0;i<l;i++)
        {
           sF+=F[i];
           if(i==0)continue;
           CF[i]=CF[i-1]+F[i];
        }
    }
    void display()
    {
        System.out.println("  x    f    cF  ");
        System.out.println("          ");
        for(int i=0;i<l;i++)
        {
           System.out.println("  "+X[i]+"    "+F[i]+"    "+CF[i]+"  ");
        }
    }
}



class Mean extends Stat
{
    Mean(int x[],int f[])
    {
        super(x,f);
    }
    int calMean()
    {
        int sFX=0;int Mean;
        for(int i=0;i<l;i++)
        {
           sFX+=X[i]*F[i];
        }
        Mean=sFX/sF;
        return Mean;
    }
}



class Mode extends Stat
{
    Mode(int x[],int f[])
    {
        super(x,f);
    }
    int findMode()
    {
        int Mode=0, Mv=0;
        for(int i=0;i<l;i++)
        {
```

```
        if(Mode<F[i]){Mode=F[i];Mv=X[i];}
      }
      return Mv;
   }
}


class Median extends Stat
{
   Median(int x[],int f[])
   {
      super(x,f);
      int C[]=new int[F.length];
   }
   double calMedian()
   {
      double Median=0;
      if(CF[CF.length-1]%2==1)
      {
         int p=(CF[CF.length-1]+1)/2;
         for(int i=0;i<CF.length;i++)
         {
            if(CF[i]>p)
            {
               Median=X[i-1];
               break;
            }
         }
      }
      else
      {
         int p1=CF[CF.length-1]/2,p2=(CF[CF.length-1]/2)+1;
         for(int i=0;i<CF.length;i++)
         {
            int a=0,b=0;
            if(i==0)continue;
            if(CF[i-1]<p1 && CF[i+1]>p1) a=X[i];
            if(CF[i-1]<p2 && CF[i+1]>p2) b=X[i];
            Median=(a+b)/2;
         }
      }
      return Median;
   }
}




interface MyInterface
{
/* All the methods are public abstract by default
* As you see they have no body
*/
public void method1();
```

```
public void method2();
}
class Demo implements MyInterface
{
/* This class must have to implement both the abstract methods
* else you will get compilation error
*/

public void method1()
{
System.out.println(&amp;quot;implementation of method1&amp;quot;);
}
public void method2()
{
System.out.println(&amp;quot;implementation of method2&amp;quot;);
}
public static void main(String arg[])
{
MyInterface obj = new Demo();
obj.method1();
}
}
```

STACKS

&

QUEUES

```
import java.util.Scanner;
class Stack
{
   int S[];
   int top;
   Stack(int s)
   {
      S=new int[s];
      top=-1;
   }
   void push(int n)
   {
      if(isOverflow())
      {
         System.out.println("Stack is full");
         return;
      }
      top++;
      S[top]=n; //S[++top]=n
   }
   int pop()
   {
      if(isUnderflow())
      {
         System.out.println("Stack is Empty");
      }
      int t=S[top];
      S[top--]=0;
      return t;
   }
   void peek()
   {
      if(isUnderflow())
      {
         System.out.println("Stack is Empty");
         return;
      }
      System.out.println(S[top]);
   }
   void traverse()
   {
      for(int i=top;i>=0;i--)
      {
         pop();
      }
   }
   boolean isOverflow()
   {
      if(top>=S.length-1) return true;
      else return false;
   }
   boolean isUnderflow()
   {
      if(top<0) return true;
      else return false;
```

```
}
public static void main()
{
   Scanner in=new Scanner(System.in);
   Stack Book=new Stack(5);
   while(true)
   {
      System.out.println((char)12);
      System.out.println("1. Push  ");
      System.out.println("2. Pop  ");
      System.out.println("3. Peek  ");
      System.out.println("4. Go through the stack  ");
      System.out.println("0. Exit  ");

      System.out.println("Enter a choice  ");
      int x=in.nextInt();

      switch(x)
      {
         case 0:
          return;

         case 1:
          System.out.println("Enter a no. to add to the stack ");
          int i=in.nextInt();
          Book.push(i);
          break;

         case 2:
          Book.pop();
          break;

         case 3:
          Book.peek();
          break;

         case 4:
          Book.traverse();
          break;
         }
         for(long i=-1000000000l;i<2000000000l;i++)
         {}
      }
   }
}
```

```
import java.util.Scanner;
class StackChar
{
   char S[];
   int top;
   StackChar(int n)
   {
      S=new char[n];
      top=-1;
   }
   void push(char c)
   {
      if(isOverflow())
      {
         System.out.println("Stack is full");
         return;
      }
      top++;
      S[top]=c; //S[++top]=s
   }
   char pop()
   {
      if(isUnderflow())
      {
         System.out.println("Stack is Empty");
         Scanner in=new Scanner(System.in);
         System.out.println("Enter Data to Stack");
         String s=in.next();
         push(s.charAt(0));
      }
      char c=S[top];
      S[top]=0;
      top--;
      return c;
   }
   char peek()
   {
      if(isUnderflow())
      {
         System.out.println("Stack is Empty");
         return ' ';
      }
      return S[top];
   }
   void traverse()
   {
      for(int i=top;i>=0;i--)
      {
         pop();
      }
   }
   boolean isOverflow()
   {
      if(top>=S.length-1) return true;
      else return false;
```

```
    }
    boolean isUnderflow()
    {
        if(top<0) return true;
        else return false;
    }
}


class InfixToPostfix
{
    String infix,postfix;
    InfixToPostfix(String exp)
    {
        infix=exp+"}";
        postfix="";
    }
    boolean isOperator(char c)
    {
        if(c=='+' ||c=='-' ||c=='*' ||c=='/' ||c=='^')
            return true;
        return false;
    }
    boolean isBracket(char c)
    {
        if(c=='(')
            return true;
        return false;
    }
    boolean lowPrecedencepop(char x,char y)
    {
        char A[]={x,y};
        for(int i=0;i<=1;i++)
        {
            char c=A[i];
            switch(c)
            {
                case '^':
                    A[i]='3';
                    break;

                case '*': case '/':
                    A[i]='2';
                    break;

                case '+': case '-':
                    A[i]='1';
                    break;

            }
        }
        int a=A[0]-'0';
        int b=A[1]-'0';
        if(b<=a) return true;
        return false;
```

```
    }
    public void toPostfix()
    {
        StackChar st=new StackChar(infix.length());
        st.push('{');
        for(int i=0;i<infix.length();i++)
        {
            char c=infix.charAt(i);
            if(isBracket(c))
            {
                st.push(c);
            }
            else if(isOperator(st.peek()) && isOperator(c))
            {
                while(isOperator(st.peek()) && lowPrecedencepop(st.peek(),c))
                {
                    postfix=postfix+st.pop();
                }
                st.push(c);
            }
            else if(isOperator(c))
            {
                st.push(c);
            }
            else if(c==')'||c=='}')
            {
                char s=' ';
                while(s!='('&&s!='{' )
                {
                    s=st.pop();
                    if(s=='('|| s=='{') continue;
                    postfix=postfix+s;
                }
            }
            else postfix=postfix+c;
        }
    }
}
```

```
import java.util.Scanner;
class Postfix_Output
{
    String postfix,postMod_;
    Postfix_Output(String exp)
    {
        postfix=exp;
        postMod_="";
    }
    void seperateCommas()
    {
        String C="";
```

```
   C=C+postfix.charAt(0);
   for(int i=1;i<postfix.length();i++)
   {
      C=C+",";
      C=C+postfix.charAt(i);
   }
   postfix=C;
}
boolean isOperator(char c)
{
   if(c=='+' ||c=='-' ||c=='*' ||c=='/' ||c=='^')
      return true;
   return false;
}
/*void replaceValues()
{
   Scanner in=new Scanner(System.in);
   postMod_=postfix;
   System.out.println("Enter Values");
   System.out.println(" ");
   for(int i=0;i<postMod_.length();i++)
   {
      if(isOperator(postMod_.charAt(i))) continue;
      if(postMod_.charAt(i)==',') continue;
      System.out.print(postMod_.charAt(i)+"= ");
      String X=in.next();
      char x=X.charAt(0);
      postMod_.replace(postMod_.charAt(i),x);
   }
}*/
void replaceValues()
{
   Scanner in=new Scanner(System.in);
   System.out.println("Enter Values");
   System.out.println(" ");
   for(int i=0;i<postfix.length();i++)
   {
      char x=postfix.charAt(i);
      if(isOperator(x) || x==',') postMod_+=x;
      else
      {
         System.out.print(x+"= ");
         String n=in.next();
         postMod_+=n;
      }
   }
}
int calculate(int x,int y, char o)
{
   int A=0;
   if(o=='+') A=x+y;
   else if(o=='-') A=x-y;
   else if(o=='*') A=x*y;
   else if(o=='/') A=x/y;
   else if(o=='^') A=(int)Math.pow(x,y);
```

```
    System.out.println("x= "+x+"  y="+y+"  A="+A);
    return A;
  }
  void solve()
  {
    for(int i=0;i<postMod_.length();i++)
    {
      System.out.println(postMod_);
      char c=postMod_.charAt(i);
      System.out.print(c+"  "+i);
      System.out.println("");
      if(c==',') continue;
      if(isOperator(c))
      {
        int x=0,y=0;
        if(i>=5 && postMod_.charAt(i-5)!=',')
        {
          if(postMod_.charAt(i-4)!=',')
            x=Integer.parseInt(postMod_.substring(i-5,i-3));
          else x=Integer.parseInt(postMod_.substring(i-6,i-4));
          System.out.println(x);
        }
        else
        {
          x=(int)postMod_.charAt(i-4);
          x-=48;
        }
        if(postMod_.charAt(i-3)!=',')
        {
          if(postMod_.charAt(i-2)!=',')
            y=Integer.parseInt(postMod_.substring(i-3,i-1));
          else y=Integer.parseInt(postMod_.substring(i-4,i-2));
        }
        else
        {
           y=(int)postMod_.charAt(i-2);
          y-=48;
        }
        if(x<=9 && y<=9)
        {
          postMod_=postMod_.substring(0,i-4)+calculate(x,y,c)+postMod_.substring(i+1);
          i-=4;
        }
        else if(y<=9)
        {
          postMod_=postMod_.substring(0,i-5)+calculate(x,y,c)+postMod_.substring(i+1);
          i-=2;
        }
        System.out.println(postMod_);
      }
    }
  }
}
```

```
import java.util.Scanner;
class QueueProto
{
   int Q[];
   int front,rear;
   QueueProto(int s)
   {
      Q=new int[s];
      front=0;
      rear=-1;
   }
   void add(int n)
   {
      if(isOverflow())
      {
         System.out.println("Stack is full");
         return;
      }
      rear++;
      Q[rear]=n; //S[++top]=n
   }
   void remove()
   {
      Scanner in=new Scanner(System.in);
      if(isUnderflow())
      {
         System.out.println("Stack is empty");
         System.out.println("Please enter a no. to add to the Stack");
         int n=in.nextInt();
         add(n);
         return;
      }
      Q[front]=0;
      front++;
   }
   private boolean isOverflow()
   {
      if(rear==Q.length-1) return true;
      else return false;
   }
   private boolean isUnderflow()
   {
      if(rear<front) return true;
      else return false;
   }
}
```

```
import java.util.Scanner;
class Queue
{
   int Q[];
   int front,rear;
   Queue(int s)
   {
      Q=new int[s];
      front=0;
      rear=-1;
   }
   void add(int n)
   {
      /*if(isOverflow())
      {
         System.out.println("Stack is full");
         return;
      }*/
      rear++;
      Q[rear]=n; //S[++top]=n
   }
   void remove()
   {
      for(int i=0;i<rear;i++)
      {
         Q[i]=Q[i+1];
      }
      Q[rear]=0;
      rear--;
   }
   public int removeR()
   {
      int n=Q[front];
      for(int i=front;i<rear;i++)
      {
         Q[i]=Q[i+1];
      }
      Q[rear]=0;
      rear--;
      return n;
   }
}
```

```
import java.util.Scanner;
class CircularQueue
{
   int Q[];
   int front,rear,space;
   CircularQueue(int s)
   {
      Q=new int[s];
      front=0;
      rear=-1;
      space=s;
   }
   void add(int n)
   {
      if(isOverflow())
      {
         System.out.println("Stack is full");
         return;
      }
      if(rear==Q.length-1) rear=0;
      else rear++;
      space--;
      Q[rear]=n; //S[++top]=n
   }
   void remove()
   {
      Scanner in=new Scanner(System.in);
      if(isUnderflow())
      {
         System.out.println("Stack is empty");
         System.out.println("Please enter a no. to add to the Stack");
         int n=in.nextInt();
         add(n);
         return;
      }
      if(front==Q.length-1)
      {
         Q[front]=0;front=0;
      }
      else Q[front++]=0;
      space++;
   }
   private boolean isOverflow()
   {
      if(space==0) return true;
      else return false;
   }
   private boolean isUnderflow()
   {
      if(space==Q.length) return true;
      else return false;
   }
}
```

LINKED LIST

```
class Node
{
   String name;
   int rn;
   Node nxt;
   Node(int R,String n)
   {
      name=n;
      rn=R;
   }
}

import java.util.*;
class LinkedList
{
   Node start;
   void List()
   {
      Node a=new Node(1,"A");
      Node b=new Node(2,"B");
      Node c=new Node(3,"C");
      Node d=new Node(4,"D");
      start=a;
      a.nxt=b;
      b.nxt=c;
      c.nxt=d;
      d.nxt=null;
   }

   void createA()
   {
      Scanner in=new Scanner(System.in);
      System.out.print("Enter the no. of nodes to be created: ");
      int n=in.nextInt();
      System.out.print("Enter Roll no. ");
      int r=in.nextInt();
      System.out.print("Enter Name ");
      String nn=in.next();
      start=new Node(r,nn);
      Node last=start;

      for(int i=2;i<=n;i++)
      {
         System.out.print("Enter Roll no. ");
          r=in.nextInt();
         System.out.print("Enter Name ");
          nn=in.next();
         Node cr=new Node(r,nn);
         last.nxt=cr;
         last=cr;

      }
   }

   void createB()
```

```
{
   Scanner in=new Scanner(System.in);
   System.out.print("Enter the no. of nodes to be created: ");
   int n=in.nextInt();
   System.out.print("Enter Roll no. ");
   int r=in.nextInt();
   System.out.print("Enter Name ");
   String nn=in.next();
   start=new Node(r,nn);
   Node cr=start;

   for(int i=2;i<=n;i++)
   {
      System.out.print("Enter Roll no. ");
       r=in.nextInt();
      System.out.print("Enter Name ");
       nn=in.next();
      cr.nxt=new Node(r,nn);
      cr=cr.nxt;
   }
}

void addNode(int r,String n)
{
   Node cr=start;
   while(true)
   {
      if(cr.nxt==null)
      {
         cr.nxt=new Node(r,n);
         return;
      }
      cr=cr.nxt;
   }
}
void displayL()
{
   Node cr=start;
   while(cr!=null)
   {
      System.out.println("Name: "+cr.name);
      System.out.println("Roll.No: "+cr.rn);
      cr=cr.nxt;
   }
}

private void displayR(Node i)
{
  if(i!=null)
  {
   System.out.println("Name: "+i.name);
   System.out.println("Roll.No: "+i.rn);
   displayR(i.nxt);
  }
}
```

```
    void displayR()
    {
        displayR(start);
    }
    void searchN(int rs)
    {
        Node cr=start;
        while(cr!=null)
        {
            if(cr.rn==rs)
            {
                System.out.println("Roll No.: "+cr.rn);
                System.out.println("Name : "+cr.name);
                return;
            }
            cr=cr.nxt;
        }
        System.out.println("Roll No. not found");
    }
}
```

```
class NodeBT
{
   int value;
   NodeBT L,R;
   NodeBT(int v)
   {
      value=v;
      NodeBT L=null;
      NodeBT R=null;
   }
}

import java.util.Scanner;
class BinaryTree
{
   NodeBT root;
   void Tree()
   {
      NodeBT A=new NodeBT(1);
      NodeBT C=new NodeBT(2);
      NodeBT D=new NodeBT(7);
      NodeBT E=new NodeBT(8);
      NodeBT F=new NodeBT(3);
      NodeBT B=new NodeBT(4);
      NodeBT G=new NodeBT(5);
      NodeBT H=new NodeBT(6);

      root=A;
      A.L=C;
      A.R=B;
      B.L=D;
      B.R=E;
      C.L=F;
      C.R=G;
      D.L=H;
   }
   void addNode(int v)
   {
      if(root==null) {root=new NodeBT(v); return;}
      NodeBT cr=root;
      while(true)
      {
         if(cr.value>v)
         {
             if(cr.L==null){cr.L=new NodeBT(v);
                          return;}
             else cr=cr.L;
         }
         if(cr.value<=v)
         {
            if(cr.R==null){cr.R=new NodeBT(v);
                        return;}
             else cr=cr.R;
         }
      }
```

```
    }
    void create()
    {
        Scanner in=new Scanner(System.in);
        System.out.print("Enter the no. of values to be entered: ");
        int n=in.nextInt();
        NodeBT cr=root;
        for(int i=0;i<n;i++)
        {
            System.out.print("Enter value: ");
            int v=in.nextInt();
            addNode(v);
        }
    }
    void display() //L+R
    {

    }
}


import java.util.Scanner;
class BinarySearchTree
{
    NodeBT root;
    void addNode(int v)
    {
        if(root==null) {root=new NodeBT(v); return;}
        NodeBT cr=root;
        while(true)
        {
            if(cr.value>v)
            {
                if(cr.L==null){cr.L=new NodeBT(v);
                               return;}
                else cr=cr.L;
            }
            if(cr.value<=v)
            {
               if(cr.R==null){cr.R=new NodeBT(v);
                              return;}
                else cr=cr.R;
            }
        }
    }
    void create()
    {
        Scanner in=new Scanner(System.in);
        System.out.print("Enter the no. of values to be entered: ");
        int n=in.nextInt();
        NodeBT cr=root;
        for(int i=0;i<n;i++)
        {
            System.out.print("Enter value: ");
            int v=in.nextInt();
```

```
            addNode(v);
        }
    }
    private void preorder(NodeBT r)
    {
        System.out.print(r.value+" , ");
        if(r.L!=null)preorder(r.L);
        if(r.R!=null)preorder(r.R);
    }
    void preorder()
    {
        preorder(root);
    }
    private void inorder(NodeBT r)
    {
        if(r.L!=null)inorder(r.L);
        System.out.print(r.value+" , ");
        if(r.R!=null)inorder(r.R);
    }
    void inorder()
    {
        inorder(root);
    }
    private void postorder(NodeBT r)
    {
        if(r.L!=null)postorder(r.L);
        if(r.R!=null)postorder(r.R);
        System.out.print(r.value+" , ");
    }
    void postorder()
    {
        postorder(root);
    }
    void display() //L+R
    {
        NodeBT cr=root;
        NodeBT last=null;
        while(true)
        {
            if(cr.L==null)
            {
                System.out.println(cr.value+" , ");

            }
        }
    }
}
```