**/\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SUM OF SMALLEST 2 DIGITS FROM GIVEN 4 NUMBERS**

package prac;

import java.util.Arrays;

class Solution1{

public int findSumeetSum(int input1,int input2, int input3,int input4)

{

 int sum=0;

 String[] input=new String[4];

 input[0]=Integer.toString(input1);input[1]=Integer.toString(input2);input[2]=Integer.toString(input3);input[3]=Integer.toString(input4);

 for(int i=0;i<input.length;i++) {

  int[] a=new int[input[i].length()];

  for(int j=0;j<a.length;j++) {

   a[j]=Integer.parseInt(input[i].substring(j, j+1));

  }

  Arrays.sort(a);

  sum=sum+(Integer.parseInt(Integer.toString(a[0])+Integer.toString(a[1])));

 }

 return sum;

}

}

 public class FIVETWOTHREE

{

 public static void main(String[] args) {

  int result;

  Solution1 s=new Solution1();

  int input1=26674;

  int input2=105;

  int input3=37943;

  int input4=95278;

  result=s.findSumeetSum(input1, input2, input3, input4);

  System.out.println(result);

 }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**STAR AND HASH \*#**

|  |
| --- |
|  |
|  | Public class Main  { |
|  | public static void main(String[] args) { |
|  | int row =4,star = 2,inc =1; |
|  | int hash,temp,sum=0; |
|  | hash=star;for(int i=1;i<=row;i++) |
|  | { |
|  | if(i!=1){ |
|  | temp=(star\*i)+(hash\*(i-1)); |
|  | } |
|  | else{ |
|  | temp=star\*i; |
|  | } |
|  | sum=temp+sum; |
|  | temp=0; |
|  | star=hash; |
|  | hash=star+inc; |
|  | } |
|  | System.out.println("sum: "+sum); |
|  | } |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**GENERATE SERIES AND FIND NTH NUMBER.**

|  |
| --- |
| { |
|  | int a=input1; |
|  | int b=input2; |
|  | int c=input3; |
|  | int d=0;; |
|  | int diff; |
|  | for(int i=4;i<=input4;i++) |
|  | { |
|  | diff=b-a; |
|  | d=c+diff; |
|  | a=b; |
|  | b=c; |
|  | c=d; |
|  | } |
|  |
|  |
|  |
|  | } |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

|  |
| --- |
|  |
|  | **JOHN,JOHHNY,JANARDHAN. ABCD PASSWORD**  { |
|  |  |
|  | String[] s={input1,input2,input3}; |
|  | String[] front=new String[3]; |
|  | String[] middle=new String[3]; |
|  | String[] end=new String[3]; |
|  |  |
|  | int l; |
|  | for(int i=0;i<3;i++) |
|  | { |
|  | l=s[i].length(); |
|  | if(l%3==2) |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)+1); |
|  | middle[i]=s[i].substring((l/3)+1,2\*(l/3)+1); |
|  | end[i]=s[i].substring(2\*(l/3)+1); |
|  | } |
|  | else if(l%3==1) |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)); |
|  | middle[i]=s[i].substring((l/3),2\*(l/3)+1); |
|  | end[i]=s[i].substring(2\*(l/3)+1); |
|  | } |
|  | else |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)); |
|  | middle[i]=s[i].substring((l/3),2\*(l/3)); |
|  | end[i]=s[i].substring(2\*(l/3)); |
|  | } |
|  | } |
|  | String o1=front[0]+front[1]+front[2]; |
|  | String o2=middle[0]+middle[1]+middle[2]; |
|  | String o3=end[0]+end[1]+end[2]; |
|  | char[] c=o3.toCharArray(); |
|  | //String s3=""; |
|  | for(int j=0;j<c.length;j++) |
|  | { |
|  |  |
|  | if(c[j]>='A' && c[j]<='Z') |
|  | { |
|  | c[j]=Character.toLowerCase(c[j]); |
|  | //s3=s3+c[j]; |
|  | } |
|  | else |
|  | { |
|  | c[j]=Character.toUpperCase(c[j]); |
|  | //s3=s3+c[j]; |
|  | System.out.println(c[j]); |
|  | } |
|  | } |
|  | String o4=new String(c);  StringBuffer sb=new StringBuffer(50) |
|  | sb.append(o1).append(o2).append(o4); |
|  | return sb.toString(); |
|  | } |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**TOKEN**

package prac;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

import java.util.TreeMap;

class findmappednames{

String result="";

public String findname\_id(int input1,int[] input2,String[] input3) {

 TreeMap<Integer, String> ts=new TreeMap<Integer,String>();

 for(int i=0;i<input2.length;i++) {

  ts.put(input2[i], input3[i]);

 }

 Set set=ts.entrySet();

 Iterator itr=set.iterator();

 int last\_num=0;

 int diff=0;

 int amt=0;

 int count=1;

  while(itr.hasNext()) {

   Map.Entry me=(Map.Entry)itr.next();

   amt=(Integer)me.getKey()-diff;

   diff=(Integer)me.getKey();

   //System.out.println(diff);

   if(count!=4) {

    if(amt==1) {

     last\_num=(Integer)me.getKey();

     count++;

    }

    else if(amt!=1) {

     count=1;

     last\_num=0;

    }

   }

   else if(count==4){

    count=1;

    break;

   }

   //System.out.println("Count="+count);

   //System.out.println("Amt="+amt);

   //System.out.println("Last\_num="+last\_num);

   //System.out.println("Diff="+diff);

  }

 if(last\_num==0) {

  result="NONE";

  return result;

 }

 else {

  for(int j=3;j>=0;j--) {

   result=result+ts.get(last\_num-j)+":";

  }

  result=result.substring(0, result.length()-1);

  return result;

 }

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class sequence\_id {

public static void main(String[] args) {

 String finalre="";

 findmappednames fmd=new findmappednames();

 int q=10;

 int[] id=new int[q];

 String[] name=new String[q];

 id[0]=55;id[1]=60;id[2]=40;id[3]=44;id[4]=49;id[5]=45;id[6]=16;id[7]=19;id[8]=47;id[9]=46;

 name[0]="bam";name[1]="sameer";name[2]="hola";name[3]="munmun";name[4]="biryani";name[5]="chicken";name[6]="momo";name[7]="chilli";name[8]="Ramesh";name[9]="Suresh";

 finalre=fmd.findname\_id(q, id, name);

 System.out.println(finalre);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**KEY AND ADDRESS**

package prac;

import java.util.Arrays;

class key\_value{

public int key\_val(int[] input1,int input2) {

 int count=1;

 int result=0;

 int key;

 int endadd=-1;

 int naddress=0;

 for(int i=0;i<input1.length;i++) {

  if(input1[i]<0) {

   endadd=i;

  }

 }

 if(endadd<=-1) {

  Arrays.sort(input1);

  result=input1[0];

 }

 else {

  while(naddress!=endadd) {

   key=Integer.parseInt((Integer.toString(input1[naddress])).substring(1,(Integer.toString(input1[naddress]).length())));

   //System.out.println("Key="+key);

   naddress=Integer.parseInt((Integer.toString(input1[naddress])).substring(0, 1));

   //System.out.println("Next Address="+naddress);

   if(count%2==0) {

    result=result-key;

    count++;

   }

   else {

    result=result+key;

    count++;

   }

   if(naddress==endadd) {

    if((count)%2==0) {

    result=result-(Integer.parseInt((Integer.toString(input1[endadd])).substring(2,(Integer.toString(input1[endadd]).length()))));

    }

    else {

    result=result+(Integer.parseInt((Integer.toString(input1[endadd])).substring(2,(Integer.toString(input1[endadd]).length()))));

    }

   }

  }

 }

 return result;

}

}

public class key\_val {

public static void main(String[] args) {

 key\_value kv=new key\_value();

 int fresult;

 int[] input1= {47,-65,51,17,29,32};

 int input2=6;

 fresult=kv.key\_val(input1, input2);

 System.out.println(fresult);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PATIENTS**

package pracprac;  
class ps{  
public String patientstring(int in1,String[] in2,int[] in3) {  
   int y=0,z=0,x=0,fl=0;  
   for(int i=0;i<in1 && fl==0;i++)  
   {  
      x=i;  
      for(int j=0;j<in1 && fl==0;j++)  
      {  
          if(in3[j]==in3[i]+1)  
          {  
              y=j;  
              for(int k=0;k<in1;k++)  
              {  
                  if(in3[k]==in3[j]+1)  
                  {  
                      z=k;  
                      fl++;  
                      //break;  
                  }  
              }  
          }  
      }  
   }  
   if(fl==0)  
   {  
    return "none";  
   }  
   else  
   {  
    return (in2[x]+":"+in2[y]+":"+in2[z]);}  
}  
}  
  
public class PRAZZ2 {  
public static void main(String[] args)   
    {  
ps p=new ps();  
String finalresult;  
    int in1=10;  
    String[] in2={"a","b","c","d","e","f","g","h","i","j"};  
    int[] in3={2,6,12,7,9,15,10,11,16,12};  
    finalresult=p.patientstring(in1, in2, in3);  
    System.out.println(finalresult);  
}  
}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

|  |
| --- |
|  |
| GETCODETHROUGHSTRINGS |
|  |  |
|  |  |
|  | String[] s=input1.split(" "); |
|  |  |
|  | int l=0; |
|  | int sum=0; |
|  | int ans=0; |
|  | int r=0; |
|  | for(int i=0;i<s.length;i++) |
|  | { |
|  | l=s[i].length(); |
|  | sum=sum+l; |
|  | } |
|  | while(sum>10) |
|  | { |
|  | r=sum%10; |
|  | sum=sum/10; |
|  | ans=sum+r; |
|  | } |
|  | return ans;  }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  | SIMPLE ENCODED ARRAY |
|  |  |
|  | int[] a=new int[input2]; |
|  | int out1; |
|  | int out2=0; |
|  | a[input2-1]=input1[input2-1]; |
|  | for(int i=input2-1;i>0;i--) |
|  | { |
|  | a[i-1]=input1[i-1]-a[i]; |
|  | //System.out.println(i); |
|  | //System.out.println(a[i]); |
|  | out2=out2+a[i]; |
|  |  |
|  | } |
|  | out2=out2+a[0]; |
|  | out1=a[0]; |
|  | //System.out.println(out1);  //System.out.println(out2); |
|  | Result r=new Result(out1,out2); |
|  | return r;  }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  | DECREASING ORDER |
|  |  |
|  |  |
|  | int max=1; |
|  | int max2=0; |
|  | int max1=0; |
|  | for(int i=0;i<input2-1;i++) |
|  | { |
|  | if(input1[i]>input1[i+1]) |
|  | { |
|  | max++; |
|  | } |
|  | else |
|  | { |
|  | if(max>1) |
|  | { |
|  | max2++; //max=count the decreasing squence |
|  | //max2=no of decreasing sequence |
|  | //max1 =max length of sequence |
|  |  |
|  | if(max1<=max) |
|  | { |
|  | max1=max; |
|  | } |
|  | } |
|  |  |
|  | max=1; |
|  | } |
|  | if(i==input2-2 && max>1) |
|  | { |
|  | max2++; |
|  | if(max1<=max) |
|  | { |
|  | max1=max; |
|  | } |
|  | } |
|  | } |
|  | Result r=new Result(max2,max1); |
|  | return r; |
|  | } |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  | STABLE\_UNSTABLE\_PASSWORD |
|  |  |
|  |  |
|  | int[] s={input1,input2,input3,input4,input5}; |
|  |  |
|  | int num=0; |
|  | int stable\_sum=0; |
|  | int unstable\_sum=0; |
|  | int c=0; |
|  | int d=0; |
|  | int e=0; |
|  | for(int i=0;i<s.length;i++) |
|  | { int[] count=new int[10]; |
|  | num=s[i]; |
|  | while(s[i]>0) |
|  | { |
|  | c=s[i]%10; |
|  | count[c]++; |
|  | s[i]=s[i]/10; |
|  | } |
|  | d=count[c]; |
|  | for(int j=0;j<10;j++) |
|  | { |
|  | if(count[j]!=0) |
|  | { |
|  | if(d==count[j]) |
|  | { |
|  | e=0; |
|  | } |
|  | Else |
|  | { |
|  | e=1; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | if(e==0) |
|  | { |
|  | stable\_sum=stable\_sum+num; |
|  | } |
|  | else |
|  | { |
|  | unstable\_sum=unstable\_sum+num; |
|  |  |
|  | } |
|  |  |
|  | } |
|  | return stable\_sum-unstable\_sum; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  | SUM OF NON PRIME INDEXES |
|  |  |
|  | int sum=0; |
|  | int prime=0; |
|  | int flag=0; |
|  | for(int i=2;i<input2;i++) |
|  | {flag=0; |
|  | for(int j=1;j<=i/2;j++) |
|  | { |
|  | if(i%j==0&&j!=1) |
|  | flag=1; |
|  | } |
|  | if(flag==0) |
|  | { |
|  | prime+=input1[i]; |
|  | System.out.println(prime); |
|  | } |
|  | } |
|  | for(int i=0;i<input2;i++) |
|  | { |
|  | sum=sum+input1[i]; |
|  | //System.out.println(sum); |
|  | } |
|  | return sum-prime; |
|  | } |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | PALINDROME REMOVE |
|  |  |
|  | // Write code here... |
|  | String reverse; |
|  | int a = 0; |
|  | String s=String.valueOf(input1); |
|  | String b; |
|  | StringBuffer x=new StringBuffer(s); |
|  | String y=String.valueOf(x.reverse()); |
|  | if(s.equalsIgnoreCase(y)) |
|  | { |
|  | a=-1; |
|  | } |
|  | else |
|  | { |
|  | for(int i=0;i<s.length();i++) |
|  | { |
|  | StringBuffer sb=new StringBuffer(s); |
|  | sb.deleteCharAt(i); |
|  | b = String.valueOf(sb); |
|  | reverse=String.valueOf(sb.reverse()); |
|  | System.out.println(reverse + " " + s); |
|  | if(reverse.equalsIgnoreCase(b)) |
|  | { |
|  | a = i; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | if(a == -1) |
|  | { |
|  | return a; |
|  | } |
|  | else |
|  | { |
|  | return Integer.parseInt(String.valueOf(s.charAt(a))); |
|  |  |
|  | } |
|  |  |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  | NAMBARIAR NUMBER |
|  |  |
|  | char[] c=input1.toCharArray(); |
|  | int[] x=new int[20]; |
|  |  |
|  | for(int i=0;i<c.length;i++) |
|  | { |
|  | x[i]=Character.getNumericValue(c[i]); |
|  | //System.out.println(x[i]); |
|  | } |
|  | List<Integer> list=new ArrayList<Integer>(); |
|  | int sum=0; |
|  | int count=0; |
|  | for(int i=0;i<x.length;i++) |
|  | { |
|  | if(x[0]%2==0 && (sum%2==0 || sum==0)) |
|  | { |
|  | sum=sum+x[i]; |
|  | count++; |
|  | } |
|  | else if(x[0]%2!=0 && (sum%2!=0 || sum==0)) |
|  | { |
|  | sum=sum+x[i]; |
|  | count++; |
|  | } |
|  | else |
|  | { |
|  | x[0]=x[count]; |
|  | count++; |
|  | list.add(sum); |
|  | sum=x[0]; |
|  | } |
|  | } |
|  | list.add(sum); |
|  | String out=""; |
|  | for(int i=0;i<list.size();i++) |
|  | { |
|  | out=out+list.get(i); |
|  | } |
|  | int z=Integer.parseInt(out); |
|  | return z; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **USER ID GENERATION** |
|  | String sn; |
|  | String ln; |
|  | String s2 =Integer.toString(input3); |
|  |  |
|  | StringBuffer sb =new StringBuffer(); |
|  | if(input1.length()>input2.length()) |
|  | { |
|  | sn=input2; |
|  | ln=input1; |
|  | } |
|  | else if(input1.length()<input2.length()) |
|  | { |
|  | sn=input1; |
|  | ln=input2; |
|  | } |
|  | else |
|  | { |
|  | String[] s={input1,input2}; |
|  | Arrays.sort(s); |
|  | sn=s[0]; |
|  | ln=s[1]; |
|  | } |
|  | sb.append(sn.charAt(sn.length()-1)); |
|  | sb.append(ln); |
|  | int n=input4; |
|  | int l=s2.length(); |
|  | sb.append(s2.charAt(n-1)); |
|  |  |
|  | sb.append(s2.charAt(l-n)); |
|  |  |
|  | String xy= sb.toString(); |
|  | char[] c=xy.toCharArray(); |
|  | for(int i=0;i<c.length;i++) |
|  | { |
|  | if(c[i]>='A' && c[i]<='Z') |
|  | { |
|  | c[i]=Character.toLowerCase(c[i]); |
|  | } |
|  | else |
|  | { |
|  | c[i]=Character.toUpperCase(c[i]); |
|  | } |
|  | } |
|  | String x=new String(c); |
|  | System.out.println(x); |
|  | return x;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  | **SUM OF dIGITS IN CYCLIC** |
|  | { |
|  | int[] arr =new int[20]; |
|  |  |
|  |  |
|  | int b= Integer.toString(input1).length(); |
|  | int l=b; |
|  |  |
|  |  |
|  | for(int i=0;i<b;i++) |
|  | { |
|  | arr[i]=input1%10; |
|  | System.out.println(arr[i]); |
|  | input1=input1/10; |
|  |  |
|  | } |
|  |  |
|  | int sum=0; |
|  | for(int i=0;i<b;i++) |
|  | { |
|  | sum=sum+l\*arr[i]; |
|  | l--; |
|  | } |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  | **MOST FREQUENT OCCURING:** |
|  |  |
|  | // Write code here... |
|  |  |
|  | { |
|  |  |
|  | int count = 0; |
|  | int maxResult = 0, outVal = 0; |
|  | TreeMap<Integer, Integer> H1 = new TreeMap<>(); |
|  | for(int i = 0; i < input2; i++){ |
|  | String n = String.valueOf(input1[i]); |
|  | for(int j = 0; j < n.length(); j++){ |
|  | int a = Integer.parseInt(String.valueOf(n.charAt(j))); |
|  | System.out.println(a); |
|  | if(H1.containsKey(a)){ |
|  | count = (Integer)H1.get(a); |
|  | H1.put(a, count+1); |
|  | } |
|  | else{ |
|  | H1.put(a, 1); |
|  | } |
|  | } |
|  | } |
|  | System.out.println(H1); |
|  | for(int k = 0; k <= 9; k++){ |
|  | if(H1.containsKey(k)){ |
|  | if(maxResult < H1.get(k)){ |
|  | maxResult = k; |
|  | } |
|  | } |
|  | } |
|  | System.out.println(maxResult); |
|  | return maxResult; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **DECREASING SEQUENCE ..** |
|  |  |
|  | { |
|  | //Write code here... |
|  | int ctr1=0; |
|  | int ctr2=0; |
|  | int d=1; |
|  | int max=0; |
|  | for(int i=0;i<input2-1;i++){ |
|  | if(input1[i]>input1[i+1]) |
|  | {ctr2++; |
|  | } |
|  | else if(input1[i+1]>input1[i]&&(ctr2!=0)){ |
|  | ctr1++; |
|  | if(ctr2>max) |
|  | max=ctr2+1; |
|  | ctr2=0; |
|  | } |
|  | } |
|  | if((input2>2)&&input1[input2-1]<input1[input2-2]) |
|  | { |
|  | ctr1++; |
|  | max++; |
|  | } |
|  | Result r=new Result(ctr1,max); |
|  | return r;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  | **SUM OF NON PRIME INDEXES** |
|  |  |
|  | int count=0; |
|  |  |
|  | int sum=0; |
|  |  |
|  | for(int i=0;i<input2;i++) |
|  |  |
|  | { |
|  |  |
|  | if(i==0 || i==1) |
|  |  |
|  | { |
|  |  |
|  | sum=input1[0]+input1[1]; |
|  |  |
|  | } |
|  |  |
|  | else if(i==2) |
|  |  |
|  | { |
|  |  |
|  | } |
|  |  |
|  | else |
|  |  |
|  | { |
|  |  |
|  | for(int j=2;j<i;j++) |
|  |  |
|  | { |
|  |  |
|  | if((i%j)==0) |
|  |  |
|  | { |
|  |  |
|  | count=1; |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  | if(count==1) |
|  |  |
|  | { |
|  |  |
|  | System.out.println(i+" "+sum); |
|  |  |
|  | count=0; |
|  |  |
|  | sum=sum+input1[i]; |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  | return sum; |
|  |  |
|  | } |
|  |  |
|  | String strNum = String.valueOf(input1); |
|  |  |
|  | StringBuffer finalStr = new StringBuffer(); |
|  |  |
|  | String temp; |
|  |  |
|  | String revStr; |
|  |  |
|  | int pos = -1; |
|  |  |
|  | for(int i = 0; i < strNum.length(); i++){ |
|  |  |
|  | // char charRem = strNum.charAt(i); |
|  |  |
|  | StringBuffer sb = new StringBuffer(strNum); |
|  |  |
|  | finalStr = sb.deleteCharAt(i);  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  | **WORLD WIDE WEB QUESTION:** |
|  |  |
|  |  |
|  | Public int |
|  |  |
|  | { |
|  | int a; |
|  | int b; |
|  | int j=0; |
|  | String s1=" "; |
|  | int[] sum=new int[100]; |
|  | int[] num=new int[100]; |
|  | String[] s=input1.toUpperCase().split(" "); |
|  | for(int i=0;i<s.length;i++) |
|  | { |
|  | a=0; |
|  | sum[i]=0; |
|  | b=s[i].length()-1; |
|  | j=0; |
|  | while(a!=b && a<b ) |
|  | { |
|  | num[j]=Math.abs(s[i].charAt(a)-s[i].charAt(b)); |
|  | sum[i]=sum[i]+num[j]; |
|  | a++; |
|  | b--; |
|  | j++; |
|  | } |
|  |  |
|  | if(a==b) |
|  | { |
|  | num[j]=s[i].charAt(a)-64; |
|  | sum[i]=sum[i]+num[j]; |
|  |  |
|  | } |
|  |  |
|  | s1 =s1.concat(Integer.toString(sum[i])); |
|  | } |
|  |  |
|  | System.out.println(s1); |
|  |  |
|  | String s2=s1.trim(); |
|  | int ans=Integer.parseInt(s2); |
|  | return ans; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | IDENTIFY POSSIBLE WORDS |
|  |  |
|  |  |
|  | { int f=0; |
|  | String ans1; |
|  | String[] s=input2.toUpperCase().split(":"); |
|  | String ans=" "; |
|  | int a=input1.indexOf('\_'); |
|  |  |
|  | String[] replaced=input2.toUpperCase().split(":"); |
|  | for(int i=0;i<s.length;i++) |
|  | { |
|  | char[] c=s[i].toCharArray(); |
|  | c[a]='\_'; |
|  | String b=new String(c); |
|  | replaced[i]=b; |
|  | if(replaced[i].equals(input1.toUpperCase())) |
|  | { |
|  | f++; |
|  | ans=ans+s[i]+":"; |
|  | } |
|  | } |
|  | if(f>0) |
|  | { |
|  | ans1=ans.substring(1,ans.length()-1); |
|  | } |
|  | else |
|  | { |
|  | ans1="ERROR-009"; |
|  | } |
|  | return ans1; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **ENCODING 3 STRINGS(JOHN,JOHNY,JANARDAN)** |
|  | **{** |
|  | String[] s={input1,input2,input3}; |
|  | String[] front=new String[3]; |
|  | String[] middle=new String[3]; |
|  | String[] end=new String[3]; |
|  |  |
|  | int l; |
|  | for(int i=0;i<3;i++) |
|  | { |
|  | l=s[i].length(); |
|  | if(l%3==2) |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)+1); |
|  | middle[i]=s[i].substring((l/3)+1,2\*(l/3)+1); |
|  | end[i]=s[i].substring(2\*(l/3)+1); |
|  | } |
|  | else if(l%3==1) |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)); |
|  | middle[i]=s[i].substring((l/3),2\*(l/3)+1); |
|  | end[i]=s[i].substring(2\*(l/3)+1); |
|  | } |
|  | else |
|  | { |
|  | front[i]=s[i].substring(0,(l/3)); |
|  | middle[i]=s[i].substring((l/3),2\*(l/3)); |
|  | end[i]=s[i].substring(2\*(l/3)); |
|  | } |
|  | } |
|  | String o1=front[0]+front[1]+front[2]; |
|  | String o2=middle[0]+middle[1]+middle[2]; |
|  | String o3=end[0]+end[1]+end[2]; |
|  | char[] c=o3.toCharArray(); |
|  | //String s3=""; |
|  | for(int j=0;j<c.length;j++) |
|  | { |
|  |  |
|  | if(c[j]>='A' && c[j]<='Z') |
|  | { |
|  | c[j]=Character.toLowerCase(c[j]); |
|  | //s3=s3+c[j]; |
|  | } |
|  | else |
|  | { |
|  | c[j]=Character.toUpperCase(c[j]); |
|  | //s3=s3+c[j]; |
|  | System.out.println(c[j]); |
|  | } |
|  | } |
|  | String o4=new String(c); |
|  | Result res=new Result(o1,o2,o4); |
|  | return res; |
|  | } |
|  |  |
|  |  |
|  | **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **USERID GENERATION:** |
|  |  |
|  | **{** |
|  | String ln; |
|  | String sn; |
|  | StringBuffer uid=new StringBuffer(); |
|  | //StringBuffer ln=new StringBuffer(); |
|  | if(input1.length()<input2.length()) |
|  | { |
|  | sn=input1; |
|  | ln=input2; |
|  | } |
|  | else if(input1.length()>input2.length()) |
|  | { |
|  | sn=input2; |
|  | ln=input1; |
|  | } |
|  | else |
|  | { |
|  | String[] s4={input1,input2}; |
|  | Arrays.sort(s4); |
|  |  |
|  | sn=s4[0]; |
|  | ln=s4[1]; |
|  |  |
|  | } |
|  | uid.append(sn.charAt(sn.length()-1)); |
|  | uid.append(ln); |
|  | uid.append(Integer.toString(input3).charAt(input4-1)); |
|  | int len=Integer.toString(input3).length(); |
|  | uid.append(Integer.toString(input3).charAt(len-input4)); |
|  |  |
|  | String s2=uid.toString(); |
|  | char[] c=s2.toCharArray(); |
|  | for(int j=0;j<c.length;j++) |
|  | { |
|  |  |
|  |  |
|  | if(c[j]>='A' && c[j]<='Z') |
|  | { |
|  | c[j]=Character.toLowerCase(c[j]); |
|  |  |
|  | } |
|  | else if(c[j]>='a' && c[j]<='z') |
|  | { |
|  | c[j]=Character.toUpperCase(c[j]); |
|  |  |
|  |  |
|  | } |
|  | } |
|  | String ans=new String(c); |
|  |  |
|  | return ans;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **ALTERNATE ADD SUB** |
|  | { |
|  | int a=input1; |
|  | int sum=input1; |
|  | int i; |
|  | if(input2==1) |
|  | { |
|  | for(i=1;a>=1;i++) |
|  | { |
|  | if(i%2==1) |
|  |  |
|  | { |
|  | a--; |
|  | sum=sum-a; |
|  |  |
|  | System.out.println(sum+" "+a); |
|  | } |
|  | else |
|  | { |
|  | a--; |
|  | sum=sum+a; |
|  |  |
|  | System.out.println(sum+" "+a); |
|  | } |
|  | } |
|  |  |
|  | } |
|  | else{ |
|  | for(i=1;a>=1;i++) |
|  | { |
|  | if(i%2==1) |
|  |  |
|  | { |
|  | a--; |
|  | sum=sum+a; |
|  |  |
|  | System.out.println(sum+" "+a); |
|  | } |
|  | else |
|  | { |
|  | a--; |
|  | sum=sum-a; |
|  |  |
|  | System.out.println(sum+" "+a); |
|  | } |
|  | } |
|  |  |
|  | } |
|  | return sum; |
|  | }  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  |  |
|  | **NAMBIAR NUMBER** |
|  | { |
|  | int z,num,sum=0,j=0; |
|  | int[] sumarr=new int[20]; |
|  | boolean flag,check; |
|  | for(int i=0;i<input1.length();i++) |
|  | { |
|  | num=(int)input1.charAt(i); |
|  | z=i-1; |
|  | if(num%2==0) |
|  | { |
|  | flag=true; |
|  | check=false; |
|  | } |
|  | else |
|  | { |
|  | flag=false; |
|  | check=true; |
|  | } |
|  | while(flag!=check && z<input1.length()-1) |
|  | { |
|  | z++; |
|  | sum=sum+input1.charAt(z)-48; |
|  | if(sum%2==0) |
|  | { |
|  | flag=true; |
|  | } |
|  | else |
|  | { |
|  | flag=false; |
|  |  |
|  | } |
|  |  |
|  | } |
|  | sumarr[j]=sum; |
|  | sum=0; |
|  | j++; |
|  | i=z; |
|  |  |
|  | } |
|  | String ans=""; |
|  | for(int k=0;k<j;k++) |
|  | { |
|  |  |
|  | ans=ans+""+sumarr[k]; |
|  | } |
|  |  |
|  | return Integer.parseInt(ans); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | DECREASING SEQUENCE |
|  |  |
|  | { |
|  | int num=input1[0]; |
|  | int z=0,max=0; |
|  | // int[] arr=new int[]; |
|  | int count=1; |
|  | if(input2>1) |
|  | { |
|  | for(int i=1;i<input2;i++) |
|  | { |
|  | System.out.println("first"+i); |
|  | if(input1[i]<num) |
|  | { |
|  | count++; |
|  | if(i==input2-1) |
|  | { |
|  | if(count>1) |
|  | { |
|  | z++; |
|  | } |
|  | if(max<count) |
|  | { |
|  | max=count; |
|  | } |
|  | } |
|  | num=input1[i]; |
|  | System.out.println(i); |
|  | } |
|  | else |
|  | { |
|  |  |
|  | if(count>1) |
|  | { |
|  | if(max<count) |
|  | { |
|  | max=count; |
|  | } |
|  | System.out.println(z+" "+count); |
|  | z++; |
|  |  |
|  | } |
|  | count=1; |
|  | num=input1[i]; |
|  |  |
|  | } |
|  | } |
|  |  |
|  |  |
|  | Result r=new Result(z,max); |
|  | return r; |
|  | } |
|  | else |
|  | { |
|  | Result r=new Result(0,0); |
|  | return r; |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | int temp=0; |
|  | int max=0; |
|  | int ans=0; |
|  | int count[]=new int[10]; |
|  | for(int i=0;i<input2;i++) |
|  | { |
|  | temp=input1[i]; |
|  | while(temp>0) |
|  | { |
|  | count[temp%10]++; |
|  | temp=temp/10; |
|  | } |
|  | } |
|  | for(int j=0;j<10;j++) |
|  | { |
|  | if(max<=count[j]) |
|  | { |
|  | max=count[j]; |
|  | ans=j; |
|  | } |
|  | } |
|  | return ans; |
|  | }  \*/ |
|  |  |
|  |  |
|  |  |