----------------------------------------------subsets----------------------------------------------------------------------

**package** practise;

**import** java.util.Scanner;

**public** **class** solution3 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//subsets

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter the number of elements in the set: ");

**int** N = sc.nextInt();

**int**[] sequence = **new** **int**[N];

**for** (**int** i = 0; i < N; i++)

sequence[i]=sc.nextInt();

System.***out***.println("The elements in the set : ");

**for** (**int** i = 0; i < N; i++)

System.***out***.print(sequence[i] + " ");

System.***out***.println("\nEnter the number of elements in the subsets: ");

**int** n = sc.nextInt();

**int** binary[]=**new** **int**[(**int**)Math.*pow*(2,N)];

**for**(**int** i=0;i< Math.*pow*(2,N);i++){

**int** b = 1;

binary[i] = 0;

**int** num = i, count = 0;

**while** (num > 0)

{

**if** (num % 2 == 1)

count++;

binary[i] += (num % 2) \* b;

num /= 2;

b = b \* 10;

}

**if** (count == n)

{

System.***out***.print("{ ");

**for** (**int** j = 0; j < N; j++)

{

**if** (binary[i] % 10 == 1)

System.***out***.print(sequence[j] + " ");

binary[i] /= 10;

}

System.***out***.println("}");

}

}

sc.close();

}

}

----------------------------------------------subsets----------------------------------------------------------------------

-----------------------------------------------------Election-------------------------------------------

package practise;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import java.util.TreeMap;

public class Solution6 {

public static void main(String[] args) {

// TODO Auto-generated method stub

//Election

Scanner in = new Scanner(System.in);

int t=in.nextInt();

HashMap<String,Integer> tm=new HashMap<String,Integer>();

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

String[] ece = new String[t];

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

{

ece[i]=in.next();

}

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

int vote = 0;

String cacdvote = ece[i];

for (int k = 0; k < ece.length; k++) {

if (cacdvote.equals(ece[k])){

vote++;

}

}

tm.put(cacdvote,vote);

}

// Collections.sort(tm,new votecomparotor());

// Collections.so

int max=0;

for(Map.Entry m:tm.entrySet()){

if(max < (int)m.getValue())

max=(int)m.getValue();

}

for(Map.Entry m:tm.entrySet()){

if(max ==(int)m.getValue())

hm.put((String) m.getKey(),(int) m.getValue());

}

String last=hm.lastKey();

// System.out.println(max);

System.out.println(last);

}}

-----------------------------------------------------Election-------------------------------------------

--------------------------------------------------Money Collection--------------------------------------

**package** practise;

**import** java.util.Scanner;

**public** **class** Solution7 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//Money Collection

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter the no of students");

**int** n=sc.nextInt();

System.***out***.println("enter the unlucky number");

**int** u=sc.nextInt();

**int** sum=0;

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

sum=sum+i;

**if**(sum==u){

sum=sum-1;

}

}

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

}

--------------------------------------------------Money Collection--------------------------------------

----------------------------------------------------minmax-----------------------------------------------------------------------

package practise;

import java.util.ArrayList;

import java.util.Scanner;

import java.util.TreeMap;

import java.util.TreeSet;

public class Solution8 {

public static void main(String[] args) {

// TODO Auto-generated method stub

//min and max

Scanner sc=new Scanner(System.in);

System.out.println("enter the no of min array");

int n=sc.nextInt();

int min[] =new int[n];

//int result[] =new int[n];

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

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

min[i]=sc.nextInt();

System.out.println("enter the no of max array");

int m=sc.nextInt();

int max[] =new int[m];

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

max[i]=sc.nextInt();

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

int temp=min[i];

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

if(temp<=max[j])

result.add(temp);

}

}

for(Integer x:result)

System.out.println(x);

Integer result1[] = (Integer[]) result.toArray();

System.out.println(result1);

}

}

------------------------------------------------------minmax-------------------------------------------------------

--------------------------------------------------VolleyBall Match-----------------------------------

package hacker;

import java.math.BigInteger;

import java.util.Scanner;

public class Solution1 {

//VolleyBall Match

static BigInteger tw = BigInteger.ONE;

static BigInteger tl = BigInteger.ONE;

static BigInteger num = BigInteger.ONE;

static BigInteger den = BigInteger.ONE;

static BigInteger v24 = BigInteger.ONE;

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int a = in.nextInt();

int b = in.nextInt();

BigInteger mo = BigInteger.valueOf(1000000007);

BigInteger way = BigInteger.ZERO;

if(a ==25 && b==24){

}else if(a<25 && b<25){

}else if((a == 25 && b<24) || (b == 25 && a<24)){

int c;

if(a == 25){

tw = BigInteger.valueOf(a-1+b);

tl = BigInteger.valueOf(b);

c = b;

}else{

tw = BigInteger.valueOf(b-1+a);

tl = BigInteger.valueOf(a);

c = a;

}

for(int i=1;i<=c;i++){

num = num.multiply(tw);

tw = tw.subtract(BigInteger.ONE);

den = den.multiply(tl);

tl = tl.subtract(BigInteger.ONE);

}

way = num.divide(den).mod(mo);

}else if((a-b==2) || (b-a==2)){

v24 = BigInteger.valueOf(801728689);

num = v24;

int c;

if(a>b){

c = b - 24;

}else{

c = a - 24;

}

tw = BigInteger.valueOf(2);

tw = tw.modPow(BigInteger.valueOf(c+1), mo);

way = num.multiply(tw).mod(mo);

}

System.out.println(way);

}

}

--------------------------------------------------VolleyBall Match-------------------------------------

-------------------------------------Fleet of Vehicles------------------------------

**package** hacker;

**import** java.util.Scanner;

**public** **class** Solution3 {

//Fleet of Vehicles

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

**int** combinations=0;

**int** number=sc.nextInt();

**if**(number%2 ==0){

**for**(**int** i=0;i<=number/2;i++){

**for**(**int** j=0;j<=number/4;j++){

**if**((2\*i+4\*j)==number)

combinations=combinations+1;

}

}

System.***out***.println(combinations);}

**else**{

combinations=0;

System.***out***.println(combinations);

}

}

}

-------------------------------------Fleet of Vehicles------------------------------

-----------------------------------------------------------Walkto remember---------------------------------

package hacker;

import java.util.HashSet;

import java.util.Set;

public class Solution5 {

//Walkto remember

static String directions = "rrlrlr";

static int length = 6;

static int starting = 1;

static int end = 2;

static int currentPoint = starting;

static Set<String> lPossiblePaths = new HashSet<>();

static StringBuffer sbCurrentPath = new StringBuffer("");

public static Set<String> generatePerm(String input) {

Set<String> set = new HashSet<String>();

if (input == "")

return set;

Character a = input.charAt(0);

if (input.length() > 1) {

input = input.substring(1);

Set<String> permSet = generatePerm(input);

for (String x : permSet) {

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

set.add(x.substring(0, i) + a + x.substring(i));

}

}

} else {

set.add(a + "");

}

return set;

}

public static void main(String[] args) {

Set<String> generatePerm = generatePerm(directions);

for (String s : generatePerm) {

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

moveTo(s.substring(i, i + 1));

}

System.out.println("Compleeted Perm s: "+ s + " paths are \n" + lPossiblePaths);

sbCurrentPath.setLength(0);

currentPoint = starting;

}

System.out.println(lPossiblePaths.size());

}

static void moveTo(String sMoveDir) {

int nextPoint = sMoveDir.equalsIgnoreCase("R") ? currentPoint + 1 : currentPoint - 1;

if ((sMoveDir.equalsIgnoreCase("R") && nextPoint <= length)

|| (sMoveDir.equalsIgnoreCase("L") && nextPoint >= starting)) {

sbCurrentPath.append(sMoveDir.equalsIgnoreCase("R") ? "R" : "L");

if ((sMoveDir.equalsIgnoreCase("R") && nextPoint == end)

|| sMoveDir.equalsIgnoreCase("L") && nextPoint == starting) {

lPossiblePaths.add(sbCurrentPath.toString());

// sbCurrentPath.setLength(0);

}

currentPoint = nextPoint;

}

// else ignore

}

}

------------------------------------Walkto remember----------------------------------

-----------------------------------movierating-----------------------------------------------------

**package** practise;

**import** java.util.Scanner;

**public** **class** Movies {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

//9,-1,-3,4,5

**int** counter=0;

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter the no of movies");

**int** n=sc.nextInt();

**int** rating=0;

**int**[] movies=**new** **int**[n];

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

movies[i]=sc.nextInt();

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

**if**(movies[i]<0)

counter=counter+1;

}

**if**(counter==movies.length){

**int** evenrating=0;

**int** oddrating=0;

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

**if**(i%2 ==0){

evenrating=evenrating+movies[i];

}

**else**

oddrating=oddrating+movies[i];

/\*\*if(movies[i]<movies[i+1]){

rating=rating+movies[i+1];

}\*\*/

}

**if**(evenrating<oddrating)

rating=oddrating;

**else**

rating=evenrating;

}

**else**{

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

**if**(movies[i]>0){

rating=rating+movies[i];

movies[i]=0;

}

}

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

**if**(movies[i]<0 && movies[i+1]<0){

**if**(movies[i]<movies[i+1])

{

rating=rating+movies[i+1];

// movies[i]=0;

}

**else**

{

rating=rating+movies[i];

//movies[i+1]=0;

}

}

}

}

System.***out***.println(rating);

}}

-------------------------------------movierating--------------------------------

-----------------------------------counter bsttree----------------------------------------

**package** day3;

**public** **class** Node {

**private** Comparable data;

**static** **int** *counter*;

**private** Node left;

**private** Node right;

**private** **static** Node *root*;

/\*\*

\* Tree constructor

\*/

**public** Node() {

*root* = **null**;

}

/\*\*

\* Node constructor

\*/

**private** Node(Comparable data) {

**this**.data = data;

left = **null**;

right = **null**;

}

/\*\*

\* Tree method to add new element

\*

\* **@param** data is a new data we want to add to a tree

\*/

**public** **void** addToTree(**int** [] a) {

*counter*=0;

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

**int** data=a[i];

**if** (*root* == **null**)

*root* = **new** Node(data);

**else**

add(*root*, data);

System.***out***.println(*counter*);

}}

// Private method that add new element

**private** **void** add(Node target, Comparable data ) {

*counter*=*counter*+1;

**if** (data.compareTo(target.data)<0) {

**if** (target.left == **null**)

target.left = **new** Node(data);

**else**

add(target.left, data);

} **else** {

**if** (target.right == **null**)

target.right = **new** Node(data);

**else**

add(target.right, data);

}

//return counter;

}

/\*\*

\* Prints out tree nodes

\*/

/\*\*public void printTree() {

print(root);

}\*/

// Private recursive method to print a tree

**private** **void** print(Node n) {

**if** (n.left != **null**)

print(n.left);

System.***out***.println(n.data);

**if** (n.right != **null**)

print(n.right);

}

**public** **static** **void** main(String[] args) {

Node tree = **new** Node();

// tree.addToTree(3);

**int** a[]={1,2,3};

tree.addToTree(a);

/\*\* tree.addToTree(2);

tree.addToTree(1);

tree.addToTree(3);

// tree.addToTree(2);

// tree.printTree();\*\*/

}

}

---------------------------------------countertree--------------------------------

-------------------------------------finaldiscount-------------------------------

**package** day3;

**import** java.util.ArrayList;

**public** **class** FinalDiscountPrice {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** sum=0;

**int** prices[]={5,4,3,2,4};

**int** nodiscountindex[]= {0,0,0,0,0};

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

**int** discount=0;

**if**(prices[i]<i+1){

sum=sum+prices[i];

nodiscountindex[i]=1;

}

**else**{

discount=prices[i+1];

sum=sum+(prices[i]-discount);

}

}

System.***out***.println(sum);

System.***out***.println("no discount items");

**for**(**int** i=0;i<nodiscountindex.length;i++)

{

**if**(nodiscountindex[i]==1)

System.***out***.println(i);

}

}

}

---------------------------------finaldiscout-------------------------------------------------------

--------------------------readingarticle-----------------------------------

**public** **class** ReadingArticles {

**public** **static** **void** main(String[] args) {

**int**[] intellectual = {1,4,6,3};

**int**[] pages = {1,2,2,3};

**int** maxInDay=8;

**int** max=0;

**for**(**int** i=0;i<pages.length-1;i++)

{

**for**(**int** j=i+1;j<pages.length;j++){

**if**((pages[i]\*2 + pages[j]\*2)<=maxInDay)

{

**if**((intellectual[i]+intellectual[j])>max)

max=intellectual[i]+intellectual[j];

}

}

}

System.***out***.println(max);

}

}

-----------------------------reading article-----------------------------

-----------sum of subsets is fixed length------------------------------------------

import java.util.Stack;

class Main1 {

/\*\* Set a value for target sum \*/

public static final int TARGET\_SUM = 15;

private Stack<Integer> stack = new Stack<Integer>();

/\*\* Store the sum of current elements stored in stack \*/

private int sumInStack = 0;

public void populateSubset(int[] data, int fromIndex, int endIndex) {

/\*

\* Check if sum of elements stored in Stack is equal to the expected

\* target sum.

\*

\* If so, call print method to print the candidate satisfied result.

\*/

if (sumInStack == TARGET\_SUM) {

print(stack);

}

for (int currentIndex = fromIndex; currentIndex < endIndex; currentIndex++) {

if (sumInStack + data[currentIndex] <= TARGET\_SUM) {

stack.push(data[currentIndex]);

sumInStack += data[currentIndex];

/\*

\* Make the currentIndex +1, and then use recursion to proceed

\* further.

\*/

populateSubset(data, currentIndex + 1, endIndex);

Integer i= (Integer) stack.pop();

//System.out.println(i);

sumInStack -= i;

//System.out.println(sumInStack);

}

}

}

/\*\*

\* Print satisfied result. i.e. 15 = 4+6+5

\*/

private void print(Stack<Integer> stack) {

StringBuilder sb = new StringBuilder();

sb.append(TARGET\_SUM).append(" = ");

for (Integer i : stack) {

sb.append(i).append("+");

}

System.out.println(sb.deleteCharAt(sb.length() - 1).toString());

}

}

public class Main {

private static final int[] DATA = { 1, 3, 4, 5, 6, 2, 7, 8, 9, 10, 11, 13,

14, 15 };

public static void main(String[] args) {

Main1 get = new Main1();

get.populateSubset(DATA, 0, DATA.length);

}

}

-----------sum of subsets is fixed length------------------------------------------