//Rotate an array of n elements to the right by k steps.   
//For example, with n = 7 and k = 3, the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4].

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** RotateArrayElements {

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

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

List<Integer> al = **new** LinkedList<Integer>();

al.add(1);

al.add(2);

al.add(3);

al.add(4);

al.add(5);

al.add(6);

al.add(7);

**int** k = 3;

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

Integer temp = al.get(0);

al.remove(temp);

al.add(temp);

}

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

}

}

// Given numRows, generate the first numRows of Pascal's triangle.   
  
**package** Assig2;

**public** **class** PascalTriangle {

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

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

**int** numRows =5;

**int** a= 1;

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

**for**( **int** j=numRows;j>i;j--){

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

}

a=1;

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

System.***out***.print(a + " ");

a = a \* (i - k) / (k + 1);

}

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

}

}

}

//Given a matrix of m x n elements (m rows, n columns), return all elements of the matrix in spiral order

**class** SpiralMatrice

{

**static** **int** *m*=4;

**static** **int** *n*=4;

**static** **int** *rowStart* =0;

**static** **int** *rowEnd* = 3;

**static** **int** *colStart* = 0;

**static** **int** *colEnd* = 3;

**public** **static** **void** main (String[] args) **throws** java.lang.Exception

{

**int**[][] a = {{1,2,3},{4,5,6},{7,8,9}};

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

*spiral*(a,result);

}

**public** **static** **void** spiral(**int**[][] a, List<Integer> result){

**while**(*rowEnd*>0 && *colEnd*>0){

*straightRow*(a,result);

*rowStart*++;

*downColumn*(a,result);

*colEnd*--;

*reverseRow*(a,result);

*rowEnd*--;

*upColumn*(a,result);

*colStart*++;

}

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

}

**public** **static** **void** upColumn(**int**[][] a, List<Integer> result) {

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

**for**(**int** i=*rowEnd*-1;i>=*rowStart*;i--){

result.add(a[i][*colStart*]);

}

}

**public** **static** **void** reverseRow(**int**[][] a, List<Integer> result) {

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

**for**(**int** i=*colEnd*-1;i>=*colStart*;i--){

result.add(a[*rowEnd*-1][i]);

}

}

**public** **static** **void** downColumn(**int**[][] a, List<Integer> result) {

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

**for**(**int** i=*rowStart*;i<*rowEnd*;i++){

result.add(a[i][*colEnd*-1]);

}

}

**public** **static** **void** straightRow(**int**[][] a, List<Integer> result) {

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

**for**(**int** i=*colStart*;i<*colEnd*;i++){

result.add(a[*rowStart*][i]);

}

}

}

// Given a collection of intervals, merge all overlapping intervals

**public** **class** Intervals {

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

/\* List list = new ArrayList();

InterSet l1 = new InterSet();

l1.setStart(1);

l1.setEnd(3);

InterSet l2 = new InterSet();

l2.setStart(2);

l2.setEnd(6);

InterSet output = new InterSet();

list.add(l1);

list.add(l2);

for (int i = 0 ; i<list.size(); i++){

if(list)

}\*/

}

}

**class** InterSet{

**int** start;

**int** end;

**public** **int** getStart() {

**return** start;

}

**public** **void** setStart(**int** start) {

**this**.start = start;

}

**public** **int** getEnd() {

**return** end;

}

**public** **void** setEnd(**int** end) {

**this**.end = end;

}

}

//Write a method to check if input String is Palindrome

**public** **class** PalindromeString {

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

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

String s = "AWA";

*isPalindrome*(s);

}

**public** **static** **void** isPalindrome(String s) {

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

StringBuffer sb = **new** StringBuffer(s);

sb.reverse();

**if**(sb.toString().equals(s)){

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

}

**else** System.***out***.println("not Palindrome");

}

}

// Write a program to print all permutations of String

**public** **class** Permutation {

**public** **static** **void** permute(String str,**int** i, **int** j ,StringBuffer sb){

**if**(i==j){

**if**(!sb.toString().contains(str)){

sb.append(str);

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

}

}

**else**{

**for**(**int** k=i;k<=j;k++){

str = *swap*(str, i, k);

*permute*(str, i + 1, j,sb);

str = *swap*(str, i, k);

}

}

}

**public** **static** String swap(String str, **int** i,**int** k){

**char**[] a = str.toCharArray();

**char** temp = a[i];

a[i] = a[k];

a[k] = temp;

str = String.*valueOf*(a);

**return** str;

}

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

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

StringBuffer sb = **new** StringBuffer("");

*permute*("AAB",0,"AAB".length()-1,sb);

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

}

}

//Given a range [m, n] where 0 <= m <= n <= 2147483647, return the bitwise AND of all numbers in this range, inclusive

**public** **class** BitWise {

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

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

**int** m,n;

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

m=sc.nextInt();

n=sc.nextInt();

**if**(m>=0 && n>=0 && n>=m && n<=2147483647){

*bitwise*(m,n);

}

**else** System.***out***.println("invalid input");

sc.close();

}

**public** **static** **void** bitwise(**int** m, **int** n) {

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

**int** c=m;

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

c = c&i;

}

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

}

//Find if there exists any pair of numbers which sum is equal to the value

**public** **class** SumOfPairOfNumbersEqualTovalue {

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

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

ArrayList<Integer> al = **new** ArrayList<Integer>();

al.add(1);

al.add(2);

al.add(3);

al.add(4);

al.add(5);

al.add(6);

al.add(7);

*find*(4,al);

}

**public** **static** **void** find(Integer k, ArrayList<Integer> al) {

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

**boolean** flag = **false**;

**for**(**int** i=0;i<al.size();i++){

Integer obj1 = al.get(i);

**for**(**int** j=0;j<al.size();j++){

**if**(j!=i){

Integer obj2 = al.get(j);

**if**(obj1+obj2 == k){

flag = **true**;

}

}

}

}

**if**(flag == **true**){

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

}

**else** System.***out***.println("false");

}

}