// 1 – You will be provided with a 2D array(n x m). You have to generate a 1D array of elements from the given matrix

// in spiral order.

// I/P

// mat = [ [1, 2, 3],

// [4, 5, 6],

// [7, 8, 9]]

// O/P

// Ans = [1, 2, 3, 6, 9, 8, 7, 4, 5]

import java.util.ArrayList;

import java.util.List;

public class SpiralMatrix {

public static void main(String[] args) {

int[][] matrix = {{1,2,3},{5,6,7},{9,10,11}};

List<Integer> res = new ArrayList<>();

res = spiral(matrix, res);

System.out.println(res);

}

public static List<Integer> spiral(int[][] mat,List<Integer> res){

int row = mat.length;

int col = mat[0].length;

int left = 0, top = 0;

int right = col-1, bottom = row-1;

while(left <= right && top <= bottom){

// ----->

for(int i=left; i<=right; i++){

res.add(mat[top][i]);

}

top++;

// Down

for(int i=top; i<=bottom; i++){

res.add(mat[i][right]);

}

right--;

// <------

for(int i=right; i>=left; i--){

res.add(mat[bottom][i]);

}

bottom--;

//Up

for(int i=bottom; i>=top; i--){

res.add(mat[i][left]);

}

left++;

}

return res;

}

}

// 2 – Given a 2D array(n x m). Rotate the array 90 degrees clockwise

// I/P

// mat = [ [1, 2, 3],

// [4, 5, 6],

// [7, 8, 9]]

// O/P

// Ans = [ [7, 4, 1],

// [8, 5, 2],

// [9, 6, 3]]

public class Rotate90Deg {

public static void main(String[] args) {

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

int[][] res = new int[mat.length][mat[0].length];

int r=0;

//Logic to get the Rotatae 90 degree matrix

for(int col = 0; col < mat[0].length; col++){

int c = 0;

for(int row = mat.length-1; row >=0; row--){

res[r][c] = mat[row][col];

c++;

}

r++;

}

for(int i=0; i<3; i++){

for(int j=0; j<3; j++){

System.out.print(res[i][j] + " ");

}

System.out.println();

}

}

}

// 3 – Consider that there is a sorted array (in non – descending order) of length n. And then it gets rotated n(1 <= n <= length of the array ) times. Now you have to find the smallest element in the array. (Rotated => removing the 1st element and adding it to the last of the array)

// I/P

// arr = [ 3, 4, 7, 9, 1, 2]

public class FindSmall {

public static void main(String[] args) {

int[] arr = {3,4,5,7,9,0,2};

System.out.println(smallest(arr));

}

public static int smallest(int[] arr){

int l = 0;

int r = arr.length - 1;

while(arr[l] > arr[r-1]){

l++;

r--;

}

return Math.min(arr[l], arr[r]);

}

}

// 5 – Decrypting the given string. The string will contain lowercase English characters, positive integers, [lowercase

// English alphabets enclosed by square brackets( “[ ]”). All the square brackets will have a positive integer before

// them. And the characters inside the “[ ]” should be printed n times continuously in the output.( Eg. If the string is

// n[abc] then the output should be abcabcabc.... (n times))

// I/P

// string = “abc2[abcd2[efg]1[ab]]abcd”

// O/P

// ans = “abcabcdefgefgababcdefgefgababcd”

import java.util.Stack;

public class DecryptString{

public static void main(String args[]){

System.out.println("Hello World");

String modString = soln("abc2[ef3[o]]c");

System.out.println(modString);

}

public static String soln(String str){

StringBuilder resStr= new StringBuilder();

Stack<String> curStr = new Stack<>();

Stack<Integer> curNum = new Stack<>();

int num = 0;

for(char c : str.toCharArray()){

if(Character.isDigit(c) ){

num = num\*10+c-'0';

}

else if(c == '['){

curStr.push(resStr.toString());

curNum.push(num);

resStr = new StringBuilder();

num = 0;

}else if(c == ']'){

String lastString = curStr.pop();

int lastNum = curNum.pop();

StringBuilder append = new StringBuilder();

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

append.append(resStr);

}

resStr = new StringBuilder(lastString + append);

}else{

resStr.append(c);

}

}

return resStr.toString();

}

}