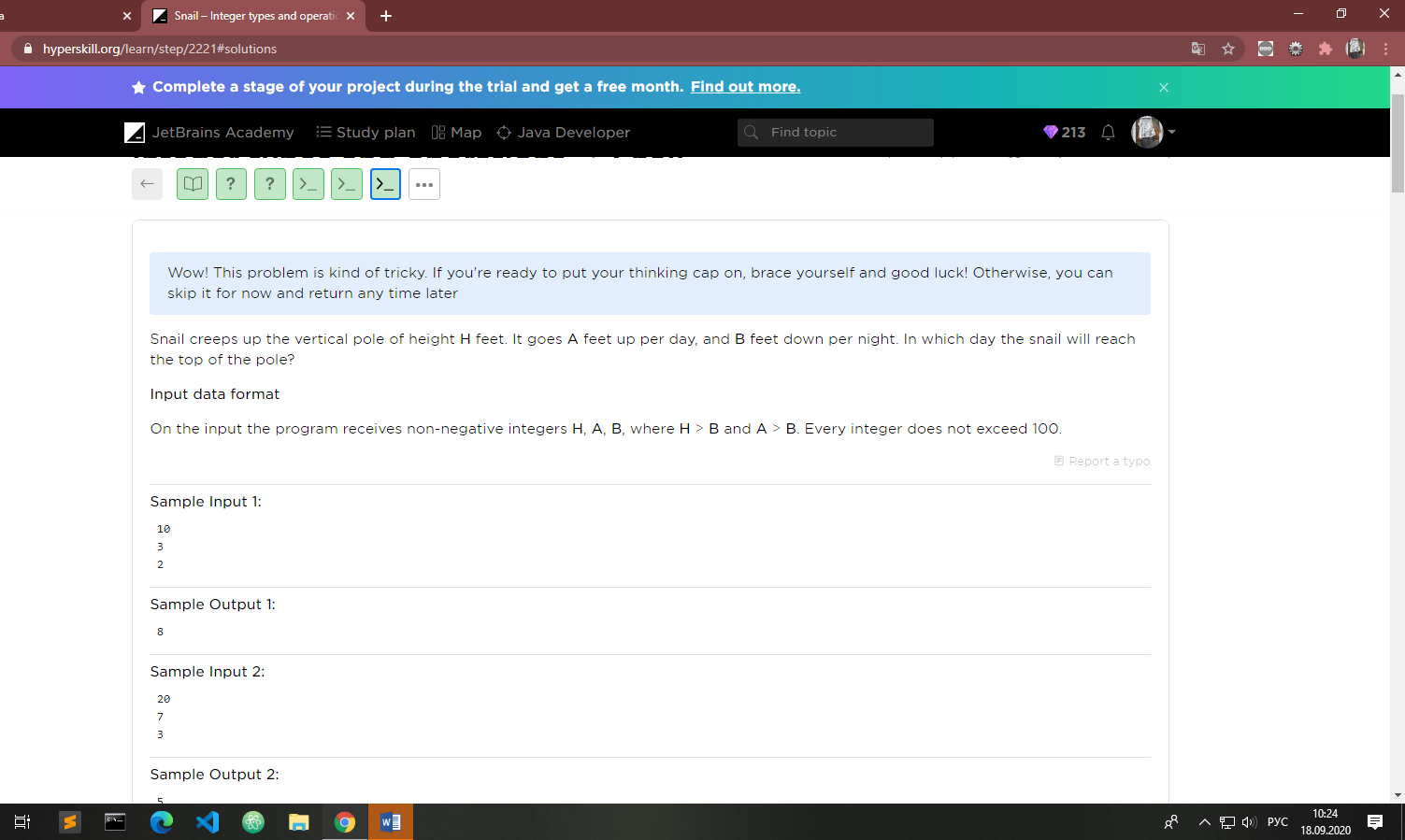
1)



2)

There are two boxes on the table. The first box has a size of X1 x Y1 x Z1, and the second box has a size of X2 x Y2 x Z2.  
You need to determine whether one of the boxes can be put inside the other. It should go in without sticking out.  
You can rotate both boxes as you want.

**Important:** two equally sized boxes cannot be placed inside one another. See the third test case as an example of how the borderline case should be treated.

Input consists of two lines:

* the first line contains numbers X1, Y1, Z1;
* the second line contains numbers X2, Y2, Z2.

All numbers are integers and greater than 0.

Output:

* "Box 1 < Box 2", if the first box can be put inside the second box ;
* "Box 1 > Box 2", if the second box can be put inside the first box;
* otherwise, output "Incompatible".

Hint

One box can be placed inside the other only if it is strictly smaller by all three dimensions. It does not necessarily mean that boxes cannot have one equal side, as you can rotate them.

 Report a typo

**Sample Input 1:**

1 2 3

5 6 4

**Sample Output 1:**

Box 1 < Box 2

**Sample Input 2:**

2 9 4

3 8 1

**Sample Output 2:**

Box 1 > Box 2

**Sample Input 3:**

1 3 7

2 8 3

**Sample Output 3:**

Incompatible

3)

A **right rotation** is an operation that shifts each element of an array to the right. For example, if an array is {1,2,3,4,5} and we right rotate it by 1, the new array will be {5,1,2,3,4}. If we rotate it by 2, the new array will be {4,5,1,2,3}. It goes like this: {1,2,3,4,5} -> {5,1,2,3,4} -> {4,5,1,2,3}.

Write a program that performs a right rotation on an array by a given number.

**Input format:**  
The first line is an array of numbers.  
The second line is the number of rotations.

**Output format:**  
Resulting array

 Report a typo

**Sample Input 1:**

1 2 3 4 5

1

**Sample Output 1:**

5 1 2 3 4

**Sample Input 2:**

1 2 3 4 5

2

**Sample Output 2:**

4 5 1 2 3

**Sample Input 3:**

1 2 3 4 5

8

**Sample Output 3:**

3 4 5 1 2

**Sample Input 4:**

11 21 1 41 51 78 90

10000

**Sample Output 4:**

41 51 78 90 11 21 1

Solution for the 3 :

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);

       String a = sc.nextLine();

       String[]b = a.split(" ");

       int [] arr = new int [b.length];

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

           arr[i] = Integer.parseInt(b[i]);

       }

        int n = arr.length-1;

        int rotA = sc.nextInt();

        int tempLast = arr[n];

        for (int i = 0; i < rotA % (n+1); i++) {

                 tempLast = arr[n];

            for (int j = n; j > 0 ; j--) {

               arr[j]=arr[j-1];

            }   arr[0]=tempLast;

        }   for(int i : arr){

            System.out.print(i + " ");

        }

    }

}

Ann put M*M* money in the bank. The bank increases Ann's deposit by P*P* percent every year. Ann wants to know how many years should pass until her deposit in the bank reaches K*K* money. Can you help her to answer this question?  
  
The input contains three integers M, P, K*M*,*P*,*K*. It is guaranteed that all numbers are positive and K \geq M*K*≥*M*.  
Output the answer to Ann's question.

 Report a typo

**Sample Input 1:**

1 100 8

**Sample Output 1:**

3

**Sample Input 2:**

100 15 120

**Sample Output 2:**

2

import java.util.\***;**public class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*)**;** double m = sc.nextInt()**;** double p = sc.nextInt()**;** double k = sc.nextInt()**;** int ctr = **0;** double money = **0;** if(m > **0** && p > **0** && k > **0**){  
  
 while (k > m){  
  
 ctr++**;** money = (m \* (p/**100**))**;** m += money**;** }  
 }  
 System.*out*.println(ctr)**;** }  
}

Wow! This problem is kind of tricky. If you're ready to put your thinking cap on, brace yourself and good luck! Otherwise, you can skip it for now and return any time later

Write a program, that takes the rectangular matrix from a sequence of lines as an input. The last line should contain the word end, indicating the end of the input.

The program should output the matrix of the same size, where each element in the position (i, j) is equal to the sum of the elements from the first matrix on the positions of their neighbors: (i-1, j)(i+1, j)(i, j-1), (i, j+1). Boundary elements have neighbors on the opposite side of the matrix.

In the case of one row or column, the element itself can be its neighbor.

 Report a typo

**Sample Input 1:**

9 5 3

0 7 -1

-5 2 9

end

**Sample Output 1:**

3 21 22

10 6 19

20 16 -1

**Sample Input 2:**

1

end

**Sample Output 2:**

4

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String aOrig = sc.nextLine();

        var a = new StringBuilder(aOrig + " ");

        int rows = 1;

        int cols = aOrig.split(" ").length;

        int[][] ard ;

        while(!"end".equals(aOrig = sc.nextLine())){

            rows++;

            a.append(aOrig + " ");

        }

        var numScan = new Scanner(a.toString().strip());

        ard = new int[rows][cols];

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

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

                ard[i][j] = numScan.nextInt();

            }

        }

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

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

               printMatrix(i,j,ard,rows,cols);

           }System.out.println();

       }

    }

    public static void printMatrix(int i , int j,int [][]ard ,int rows,int cols){

            int res = 0;

            res += ard[posRep(i - 1,rows)][j];

            res += ard[posRep(i + 1,rows)][j];

            res += ard[posRep(i,rows)][posRep(j-1,cols)];

            res += ard[posRep(i,rows)][posRep(j+1,cols)];

            System.out.print(res +" ");

    }

    public  static  int posRep(int pos,int size){

        if (pos < 0){

            return  size -1;

        } else if (pos >= size) {

            return  0;

        } else {

            return  pos;

        }

    }

}

Wow! This problem is kind of tricky. If you're ready to put your thinking cap on, brace yourself and good luck! Otherwise, you can skip it for now and return any time later

The cinema has **n** rows, each row consists of **m** seats (**n** and **m** do not exceed 20). The two-dimensional matrix stores the information on the sold tickets, number **1** means that the ticket for this place is already sold, the number **0** means that the place is available. You want to buy **k** tickets to the neighboring seats in the same row. Find whether it can be done.

**Input data format**

On the input, the program gets the number of **n** rows and **m** seats. Then, there are n lines, each containing **m** numbers (**0** or **1**) separated by spaces. The last line contains a number **k**.

**Output data format**

The program should output the number of the row with **k** consecutive available seats. If there are several rows with **k** available seats, output the first row with these seats. If there is no such a row, output the number 0.

 Report a typo

**Sample Input 1:**

3 4

0 1 0 1

1 1 0 1

1 0 0 1

2

**Sample Output 1:**

3

**Sample Input 2:**

3 3

0 1 0

1 0 0

1 1 1

3

**Sample Output 2:**

0

import java.util.\***;**class Main{  
 public static void main (String[]args){  
 Scanner sc = new Scanner(System.*in*)**;** int row = sc.nextInt()**;** int col = sc.nextInt()**;** int [][]sits = new int[row][col]**;** int ctr = **0;** for(int i =**0;** i<row **;** i++){  
 for (int j = **0;** j < col **;**j++){  
 sits[i][j] = sc.nextInt()**;** }  
 }  
 System.*out*.println()**;** int tk = sc.nextInt()**;** for(int i = **0;** i<row**;**i++){  
 ctr=**0;** for(int j = **0;** j<col **;** j++){  
 ctr++**;** if(sits[i][j]==**1**)ctr=**0;** if(ctr == tk) {  
 System.*out*.print(i + **1** + " ")**;** i = row - **1;** break**;** }  
  
  
 }  
  
 }  
  
 if(ctr<tk){  
 System.*out*.println(**0**)**;** }  
  
  
  
  
 }  
}