EASY LEVEL 4

1

Rabbit Peter

1. **Problem statement:**

Rabbit Peter has been interested in arrays since his childhood. Now he is learning arrays of length N that contain only integers from 1 to N. He was never good at math though. Yesterday he tried to figure out how many awesome arrays they were. Peter believes that an array is awesome if one of the two statements is true:

1. each element, starting from the second, is not greater than the previous

2. each element, starting from the second, is not less than the previous

Peter came to ask for help from Stewie and Brian. But they just laughed and said that the answer is too simple. Help the rabbit find the answer.

2. Input Format:

A number N - the size of the array

3. Output Format:

The number of awesome arrays by modulo 1000000007

4. Constraints:

$1 \le N \le 100000$

5. Samples Input/Output:

	Input data	Output data
1	2	4
2	3	17

6. Test Cases:

	Input data	Output data
1	12	2704144
2	19	345263536
3	20	846527841
4	26	529476652
5	35	358906180
6	38	917151454

7	42	769030659

7. Sample solution (Java):

```
import java.util.*;

import java.io.*;

import java.math.BigInteger;

public class arregloduro{

public static void main(String args[]) throws IOException{

BufferedReader lector = new BufferedReader(new
InputStreamReader(System.in));

long dp[] = new long[100001];

dp[1]=1L;

long mod = 1000000007;

for(long n = 2;n<dp.length;n++)dp[(int)n] = (((2L*((2L*n-
1)*(dp[(int)n-1]+n-1))%mod)*inv(n))%mod-n)%mod;

int tmp = Integer.parseInt(lector.readLine());

System.out.println(dp[tmp]);

}

public static long inv(long a){

return Long.parseLong(""+(new BigInteger(""+a).modInverse(new
BigInteger("1000000007"))))%1000000007;
```

```
}

}
```

2

Reca Screens

1. **Problem statement:**

Reca company produces screens. The most popular model is AB999 with the size of A × B centimeters. Due to the nature of production, screen sizes are expressed in a whole number of centimeters. Recently everybody has started buying screens with the X:Y ratio. The company wants to reduce its AB999 screen dimensions to make the X:Y ratio with the maximum possible area. Your task is to find the size of the reduced model of the screen or to find out that this is not possible.

2. Input Format:

A, *B*, *X, Y*

3. Output Format:

If a solution exists, output two positive numbers - the screen size of the new model. If there is no solution then output 0 0.

4. Constraints:

$1 \le A, B, X, Y \le 2000000000$

5. Samples Input/Output:

	Input data	Output data
1	800 600 4 3	800 600
2	1920 1200 16 9	1920 1080

6. Test Cases:

	Input data	Output data
1	1 1 1 2	0 0
2	1002105126 227379125 179460772 1295256518	0 0
3	248228385 1458744978 824699604 1589655888	206174901 397413972
4	511020182 242192314 394753578 198572007	394753578 198572007
5	134081812 857875240 82707261 667398699	105411215 850606185
6	721746595 799202881 143676564 380427290	287353128 760854580
7	912724694 1268739154 440710604 387545692	881421208 775091384

7. Sample solution (Java):

```
import java.util.Scanner;
```

```
public class Main {



    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int a = sc.nextInt();

        int b = sc.nextInt();

        int x = sc.nextInt();

        int y = sc.nextInt();

        int d = x, e = y;

        int c = d % e;

        while(c != 0) {

            d = e;

            e = c;

            c = d % e;

            if(c == 0) {

                break;

            }

        }
```

```
        x = x / e;

        y = y / e;

        if(a / x > b / y) {

            System.out.println(b/y*x + " " + b/y*y);

        } else {

            System.out.println(a/x*x + " " + a/x*y);

        }


    }


}
```

3

Seven Friendly

1. **Problem statement:**

You are given a number *a*, it contains digits 1, 6, 8, 9. Rearrange the digits in it so that a_1 (the resulting number) mod 7 == 0 (a_1 % 7 == 0)

The given number never contains leading zeros and always contains 1, 6, 8, 9 and possibly other numbers. The resulting number should not contain leading zeroes either.

2. Input Format:

The first line contains a positive integer *a* the decimal form of which contains from 4 to 10^6 digits.

3. Output Format:

Print out the resulting number that does not contain leading zeros. If it's not possible to rearrange print 0.

4. Constraints:

-

5. Samples Input/Output:

	Input	Output
1	18906	18690
2	2419323689	1698243239

6. Test cases

	Input	Output
1	16893	81963
2	16894	19684
3	16899	16989
4	11689	16198
5	91111168	11111968
6	6198	1869
7	1689999999999	9999999991968
8	9883291673084	2334788919680

7. Solution (Java)+

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class SampleClass {
public static void main(String[] args) throws java.io.IOException {
BufferedReader input = new BufferedReader(new InputStreamReader( System.in));
final int[] mods = {
1869,
1968,
1689,
6891,
1698,
1986,
8196
};
final int[] tens = {
1,
3,
2,
6,
4,
5
};
StringBuilder num = new StringBuilder(input.readLine());
num.deleteCharAt(num.indexOf("1")).deleteCharAt(num.indexOf("6"))
.deleteCharAt(num.indexOf("8")).deleteCharAt(num.indexOf("9"));
int rem = 0;
for (char c: num.toString().toCharArray()) {
rem *= 3;
```

```
rem += (c - '0');

rem %= 7;

}

int offset = tens[num.toString().length() % 6];

for (int i = 0; i < 7; i++) {

if ((i * offset + rem) % 7 == 0) {

System.out.println(mods[i] + num.toString());

return;

}

}

}

}
```

4

Sherlock

1. **Problem statement:**

Sherlock is looking at the city's crime map, represented by an $n \times m$ rectangular table. With the symbol "*" he marked three places where crimes happened. He thinks that the next crime will be in such a place, that the four places will form a rectangle parallel to the edges of map.

You need to find the coordinates of the new crime place in the matrix.

2. Input Format:

n and *m* — the number of rows and columns

Each of the next *n* lines contains *m* characters: "." - a dot means a place with no crime, and "*" - a place where a crime has been comitted.

3. Output Format:

The number of row and the number of column where the next crime will happen.

4. Constraints:

$2 \le n, m \le 100$

5. Samples Input/Output:

	Input data	Output data
1	3 2 .* .. **	1 1
2	3 3 *.* *.. ...	2 3

6. Test Cases:

	Input data	Output data
1	7 2 .. ** .. ..*	7 1

2	8 8 …….. .*…… …….. …….. …….. …….. .*…..* ……..	2 8
3	6 10 …….. …….. .*…….. …….. .*….*… ……..	3 7
4	2 10 *……*.. …….*..	2 1
5	50 4 …. …. …. …. .*.* …. …. …. …. …. …. …. ….	49 4

	....	
6	4 4 *..* ....*	3 1
7	2 3	2 1

	. ..*	

7. Source (Java):

```
import java.util.Scanner;



public class Main {

    public static void main(String args[]){

        Scanner input = new Scanner(System.in);

        int n = input.nextInt();

        int m = input.nextInt();

        char a[][] = new char[n][m];

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

            String str = input.next();

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

                a[i][j] = str.charAt(j);

            }

        }
```

```
int count = 0;

int x = 0,y = 0;

boolean found = false;

//horizontal check

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

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

        if(a[i][j] == '*'){

            ++count;

        }

    }

    found = count == 1 ? true : false;

    if(found){

        x = i;

        found = false;

        //break;

    }

    count = 0;

}
```

```
        //vertical check
        for(int i = 0;i < m;i++){
            for(int j = 0; j < n;j++){
                if(a[j][i] == '*'){
                    ++count;
                }
            }
            found = count == 1 ? true : false;
            if(found){
                y = i;
                found = false;
                //break;
            }
            count = 0;
        }

        System.out.println((x+1) + " " + (y+1));
    }
}
```

5

Simple Array Task

1. **Problem statement:**

You are given an array of positive integer elements a. Find three different indices i,j,k that ai = aj + ak.

2. Input Format:

The first line contains an integer n - the number of elements in the array.

The second line contains n integers - the elements of the array.

3. Output Format:

Output three integers i,j,k that ai = aj + ak or -1 if they don't exist. If there are multiple solutions, print out any of them.

4. Constraints:

$3 \le n \le 100$

$1 \le ai \le 1000$

5. Samples Input/Output:

	Input	Output
1	5 1 2 3 5 7	3 2 1
2	5 1 8 1 5 1	-1

6. Test cases

	Input	Output
1	4 303 872 764 401	-1

2	6 86 402 133 524 405 610	6 4 1
3	10 858 972 670 15 662 114 33 273 53 310	2 6 1
4	8 217 779 418 895 996 473 3 22	5 1 2
5	3 907 452 355	-1
6	10 983 748 726 406 196 993 2 251 66 263	-1
7	9 933 266 457 863 768 257 594 136 145	-1
8	20 551 158 517 475 595 108 764 961 590 297 761 841 659 568 82 888 733 214 993 359	13 1 6

7. Solution (Java)

```
import java.io.*;

import java.util.*;

public class SampleClass {

public static void main(String a[]) throws IOException {

BufferedReader b = new BufferedReader(new InputStreamReader( System.in));

int k = 0, j = 0, p = 0, i = 0;

String s;
```

```
p = Integer.parseInt(b.readLine());

int d[] = new int[p];

s = b.readLine();

StringTokenizer c = new StringTokenizer(s);

for (i = 0; i < p; i++)

d[i] = Integer.parseInt(c.nextToken());

for (i = 0; i < p - 1; i++) {

for (j = i + 1; j < p; j++) {

for (k = 0; k < p; k++) {

if (d[i] + d[j] == d[k]) {

System.out.print((k + 1) + " " + (j + 1) + " " + (i + 1));

System.exit(0);

}

}

}

}

System.out.print("-1");

}

}
```

6

Socks

1. **Problem statement:**

James always takes 4 pairs of socks with him for a business trip , all pairs should be of a different color. James has 4 pairs left, but maybe some of them have the same color. Therefore, he needs to go to a store and buy a few extra pairs of socks.

In order to save his money James wanted to spend as little money as possible, so you need to help James and determine the minimum number of pairs of socks he needs to buy, so that he could take 4 pairs of socks of different colors with him.

2. Input Format:

The first line contains four integers s_1, s_2, s_3, s_4 - the colors of the pairs of socks available to James.

Assume that all the possible colors are numbered with integers.

3. Output Format:

Print out a single integer - the minimum number of pairs of socks that James needs to buy.

4. Constraints:

$1 \le s_1, s_2, s_3, s_4 \le 10^9$

5. Samples Input/Output:

	Input data	Output data
1	1 7 3 3	1
2	7 7 7 7	3

6. Test cases

	Input data	Output data
1	81170865 673572653 756938629 995577259	0
2	3491663 217797045 522540872 715355328	0
3	251590420 586975278 916631563 586975278	1
4	259504825 377489979 588153796 377489979	1
5	652588203 931100304 931100304 652588203	2
6	391958720 651507265 391958720 651507265	2
7	90793237 90793237 90793237 90793237	3

7. Solution (Java)

```
import java.util.*;

public class Socks {

    public static void main(String[] args) {

        Scanner kbd = new Scanner(System.in);

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

        for(int i = 0;i<4;i++) {

            s.add(kbd.nextInt());

        }



        System.out.println(4-s.size());

    }

}
```

7

Steps

1. **Problem statement:**

Misha wants to climb a ladder consisting of n steps. In one step he can climb one or two steps. Thus, Misha wants the number of steps to be divisible by m.

What is the minimum number of steps he would need to take to climb the ladder while satisfying the mentioned condition?

2. Input Format:

The only line contains two space-delimited integers - n, m .

3. Output Format:

Print a single number - the minimum number of steps evenly divisible by m. If there is no way to climb the stairs while satisfying the condition specified, output - 1.

4. Constraints:

$0 < n \leq 10000$, $1 < m \leq 10$

5. Samples Input/Output:

	Input data	Output data
1	10 2	6
2	3 5	-1

In the first example Misha can climb the stairs for 6 turns, performing the following steps: {2, 2, 2, 2, 1, 1}.

In the second example there are only three suitable sequences of steps {2, 1}, {1, 2}, {1, 1, 1} of length 2, 2, and 3 respectively. All these numbers are not multiples of 5.

6. Test Cases:

	Input data	Output data
1	29 7	21
2	2 2	2

3	1 2	-1
4	9999 9	5004
5	20 10	10
6	9999 2	5000
7	20 10	10

7. Sample solution (Java):

```
import java.util.*;

public class Q1 {

public static void main(String[] args ) {

Scanner in = new Scanner(System.in);

int n = in.nextInt(), m = in.nextInt();

int x, y;

int a=0;

int res=-1;

while(n-(++a)*m>=0) {

y = n - a*m;

x = a*m - y;

if(x>=0&&y>=0) {

res = a*m;

break;

}

}

System.out.println(res);

}

}
```

8

The Only Way is Up

1. **Problem statement:**

The sequence *a* is called rising if for each *i* except 0 *a*i > *a*i-1 (strictly greater).

Given a sequence *x* and a positive integer *b,* you need to make x a rising sequence by adding b to one of its elements.

How many times do you need to add b to make x a rising sequence?

2. Input Format:

The first line contains two integers *n* and *d* . The second line contains the sequence elements *x*0, *x*1, ..., *xn* - 1.

3. Output Format:

Output the number of times you need to add b to one of the elements to make x a rising sequence?

4. Constraints:

$2 \le n \le 2000, 1 \le d \le 10^6$

$1 \le xi \le 10^6$

5. Samples Input/Output:

	Input	Output
1	4 2 1 3 3 2	3
2	2 1 1 1	1

6. Test cases

	Input	Output
1	2 1	0

	2 5	
2	2 1 1 2	0
3	2 7 1 1	1
4	3 3 18 1 9	10
5	3 3 10 9 12	2
6	10 3 2 1 17 10 5 16 8 4 15 17	31
7	10 3 6 11 4 12 22 15 23 26 24 26	13
8	11 4 529 178 280 403 326 531 671 427 188 866 669	473

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

int n = s.nextInt(), d = s.nextInt();

long[] a = new long[2005];

for (int i = 1; i <= n; i++) a[i] = s.nextLong();
```

```
long ans = 0;

for (int i = 2; i <= n; i++) {

if (a[i] > a[i - 1]) continue;

ans += (a[i - 1] - a[i]) / d + 1;

a[i] = a[i] + (a[i - 1] - a[i]) / d * d + d;

}

System.out.println(ans);

s.close();

}

}
```

9

Tickets to the movies

1. **Problem statement:**

The price for one movie show at the cinema is $1. Every adult can bring no more than one child with them for free. This means that an adult who brings k (k> 0) children pays $k: one ticket for themselves and (k - 1) tickets for their children. Also, an adult can go without children, in this case, they pay just one dollar.

Children cannot go to the cinema without an adult.

Calculate the minimum and maximum total dollar amount paid by the visitors.

2. Input Format:

The input consists of a single line in which two space-delimited integers n and m are given - the number of adults and the number of children at the cinema, respectively.

3. Output Format:

If n adults and m children were able to visit the cinema, on a single line print two numbers separated by a space - the minimum and maximum possible price for their visit, respectively.

Otherwise, output «Impossible» (without quotes).

4. Constraints:

$0 \le n, m \le 10^5$

5. Samples Input/Output:

	Input data	Output data
1	1 2	2 2
2	0 5	Impossible

6. Test cases

	Input data	Output data
1	2 2	2 3
2	2 7	7 8
3	4 10	10 13
4	6 0	6 6
5	7 1	7 7
6	0 0	0 0
7	71 24	71 24

7. Solution (Java)

```
import java.math.BigInteger;

import java.util.*;
```

```
public class Main {



    public static void main(String[] args) throws Exception{

        Scanner sc = new Scanner(System.in);

        int n=sc.nextInt();

        int m=sc.nextInt();

        if(n==0 && m!=0){

            System.out.println("Impossible");

            return;

        }

        int min = m<=n?n:m;

        int max = m+n-1;

        if(m==0) max=min=n;

        System.out.println(min+" "+max);

    }

}
```

Treasure Map

1. **Problem statement:**

Ivan received a treasure map from his father. The map is a matrix NxN, representing an "islands map". An "island" is a group of cells (or one unit) surrounded by zeros (or edges of the matrix) from all sides.

Cells belong to one "island" if you can move from one to another horizontally or vertically moving along non-zero cells.

The numbers on the non-zero cells represent the probability to find a treasure there. So Ivan wants to know the number of islands on the map and he needs to find the Island with the maximum total probability to find a treasure.

2. Input Format:

The first line: N - the number of rows & columns

Other lines: the NxN matrix

3. Output Format:

total number of islands, island with maximum total probability.

4. Constraints:

-

5. Samples Input/Output:

	Input data	Output data
1	5 2 0 4 1 10 0 0 0 0 0 4 2 2 0 6 0 6 0 0 7 0 0 0 2 0	5 15

2	6 2 6 0 0 0 0 3 4 0 6 0 0 0 0 4 5 0 0 0 0 3 1 0 0 0 0 0 0 1 10 0 0 0 0 1 1	3 19

6. Test Cases:

	Input data	Output data
1	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 14 0 16 0 0 1 1 0 0 0 1 0 0 1 1 0 0 2 1 0 0 0 1 0 4 1 0 0 0 0 2 0 2 0 0 0 0 0 3 0 1 9	4 19
2	2	1 1

	1 0 0 0	
3	3 0 0 0 0 0 0 0 0 0	0 0
4	3 0 0 0 0 1 1 0 0 0	1 2
5	1 1	1 1
6	4 0 1 3 4 0 1 2 0 0 1 0 0 0 1 0 0	1 13
7	2 1 1 1 1	1 4

7. Sample solution (Java):

```
public static int getIslandSum(int[][] array,int i, int j)

{

   if ((i < 0) || (j < 0) || (i == N) || (j == N) || (array[i][j] ==
0)) {

   return 0;

   }



int value = array[i][j];

array[i][j] = 0;

return value + getIslandSum(array,i+1,j) + getIslandSum(array,i-1,j)
+ getIslandSum(array,i,j+1) + getIslandSum(array,i,j-1);

}




int count = 0;

int maxSum = 0;

   for (int i = 0; i < N; i++) {
```

```
      for (int j = 0; j < N; j++) {
        if (a[i][j] != 0) {
           count++;
           int sum = getIslandSum(a,i,j);
           if (sum > maxSum) {
                maxSum = sum;
           }
        }
     }
 }
```

11

Young builder

1. **Problem statement:**

Little Bob has recently received a ‘Young Builder’ set as a gift from his older brother. This set consists of a few wooden bars, the length of each is known. The bars can be laid on top of one another if the bars are of the same length.

Bob wants to use all the bars to build the minimum number of towers. Help Bob optimally arrange the bars.

2. Input Format:

The first line contains an integer N - the number of bars available to Bob. The second line contains N integers l_i - the bars’ length.

3. Output Format:

Print out two numbers - the greatest height of the towers and their total number. Remember that Bob should use all available bars.

4. Constraints:

$1 \le N \le 10^3$

All bars lengths are integers not exceeding 10^3.

5. Samples Input/Output:

	Input	Output
1	3 1 2 3	1 3
2	4 6 5 6 7	2 3

6. Test cases

	Input	Output
1	4 3 2 1 1	2 3
2	3 20 22 36	1 3
3	1 1	1 1
4	25 47 30 94 41 45 20 96 51 110 129 24 116 9 47 32 82 105 114 116 75 154 151 70 42 162	2 23
5	5 1 1000 1000 1000 1000	4 2

6	5 1000 1000 1000 8 7	3 3
7	1 1000	1 1
8	5 5 5 5 5 5	5 1

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

int a[] = new int[1001];

int max = 1;

int count = 0;

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

a[input.nextInt()]++;

for (int j = 1; j < 1001; j++) {

if (a[j] == 0) continue;

max = Math.max(max, a[j]);

count++;

}

System.out.println(max + " " + count);

}

}
```

12

Alice and Bob and the Set

1. **Problem statement:**

Alice and Bob are playing a game with the following rules:

1) They pick some set of n integer values and take turns.

2) Each turn a player picks two numbers ai and aj from the set so that the absolute difference |ai - aj| isn't present in the set and adds |ai - aj| to the set.

3) If a player is unable to take their turn (2) they lose.

Alice takes the first turn (step). Find out who wins if both players play perfectly.

2. Input Format:

The first line contains an integer n - the number of elements in the set.

The second line contains n integers - the elements of the set.

3. Output Format:

Output the name of the winner "Alice" or "Bob", note that Alice takes the first turn (step).

4. Constraints:

$1 \le n \le 100$

$1 \le ai \le 10^9$

5. Samples Input/Output:

	Input	Output
1	2 2 3	Alice
2	2 5 3	Alice

6. Test cases

	Input	Output
1	3 5 6 7	Bob
2	10 72 96 24 66 6 18 12 30 60 48	Bob
3	10 78 66 6 60 18 84 36 96 72 48	Bob
4	2 2 6	Alice
5	10 1 999999999 999999998 999999997 999999996 999999995 999999994 999999993 999999992 999999991	Alice
6	3 4 12 18	Bob
7	4 2 3 15 30	Bob
8	2 10 4	Alice

7. Solution (Java)

```
import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.HashSet;
```

```
import java.util.Scanner;

public class SampleClass {

static int gcd(int a, int b) {

return (b == 0 ? a : gcd(b, a % b));

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

int array[] = new int[n];

int max = 0;

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

array[i] = scan.nextInt();

max = Math.max(array[i], max);

}

Arrays.sort(array);

int d = array[0];

for (int i = 1; i < n; i++)

d = gcd(array[i], d);

int rounds = (max / d) - n;

if (rounds % 2 == 0) {

System.out.println("Bob");

} else {

System.out.println("Alice");

}

}

}
```

13
Array Distances

1. **Problem statement:**

Given an array of n elements, for each element you need to find out the minimum and maximum distance to another elements. The distance is the absolute difference between two elements.

The given array is always sorted in ascending order and all of its elements are distinct.

2. Input Format:

The first line of the input contains an integer n — the number of elements in the given array.

The second line contains n distinct integers - the elements of the array in ascending order.

3. Output Format:

Print n lines each representing the minimum and maximum distance for each element. Retain the order of elements - the first element of the array should be printed on the first line, and so on.

4. Constraints:

$2 \le n \le 10^5$

$-10^9 \le xi \le 10^9$

5. Samples Input/Output:

	Input	Output
1	4 -1 0 1 3	1 4 1 3 1 2 2 4
2	3 -1000000000 0 1000000000	1000000000 2000000000 1000000000 1000000000 1000000000 2000000000

6. Test cases

	Input	Output
1	5 -2 -1 0 1 3	1 5 1 4 1 3 1 3 2 5
2	4 -5 -2 2 7	3 12 3 9 4 7 5 12
3	2 -1 1	2 2 2 2
4	3 -1 0 1	1 2 1 1 1 2
5	3 -10000 1 10000	10001 20000 9999 10001 9999 20000
6	10 1 10 12 15 59 68 130 912 1239 9123	9 9122 2 9113 2 9111 3 9108 9 9064 9 9055 62 8993 327 8211

		327 7884 7884 9122
7	2 -1000000000 1000000000	2000000000 2000000000 2000000000 2000000000

7. Solution (Java)

```
import java.util.*;

import java.math.*;

public class Main {

public static void main(String args[]) {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

int[] a = new int[n];

for (int i = 0; i < n; ++i) a[i] = in .nextInt();

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

int max, min;

if (i == 0) {

min = a[1] - a[0];

max = a[n - 1] - a[0];

} else

if (i == n - 1) {

min = a[n - 1] - a[n - 2];

max = a[n - 1] - a[0];

} else {

min = Math.min(a[i] - a[i - 1], a[i + 1] - a[i]);

max = Math.max(a[i] - a[0], a[n - 1] - a[i]);

}
```

```
System.out.println(min + " " + max);

}

in .close();

}

}
```

14

Chess Robot

1. **Problem statement:**

Garry invented a new chess figure called Robot. Robot can move (and attack) one step horizontally or vertically. So if the robot is in the cell (x, y), it can attack (x, y+1), (x, y-1), (x+1, y), (x-1, y) cells.

Gary wants to know how many robots can be placed on a chessboard n × n, so that no robot can attack any other robots.

2. Input Format:

A positive integer N

3. Output Format:

The first line - the maximum number of robots.

The next N lines - N symbols, if a cell is empty print ".", otherwise print "C".

4. Constraints:

$1 \le N \le 1000$

5. Samples Input/Output:

	Input data	Output data
1	2	2

		C. .C
2	5	5 C.C .C. C.C

6. Test Cases:

	Input data	Output data
1	4	8 C.C. .C.C C.C. .C.C
2	10	50 C.C.C.C.C. .C.C.C.C.C C.C.C.C.C. .C.C.C.C.C C.C.C.C.C. .C.C.C.C.C C.C.C.C.C. .C.C.C.C.C C.C.C.C.C. .C.C.C.C.C

3	15	113 C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C .C.C.C.C.C.C.C. C.C.C.C.C.C.C.C
4	1	1 C
5	3	5 C.C .C. C.C
6	6	18 C.C.C. .C.C.C C.C.C. .C.C.C C.C.C. .C.C.C

7	7	25 C.C.C.C .C.C.C. C.C.C.C .C.C.C. C.C.C.C .C.C.C. C.C.C.C

7. Sample solution (Java):

```
import java.util.Scanner;



public class Main {

public static void main(String... args) {



Scanner sc = new Scanner(System.in);

int n = sc.nextInt();



System.out.println((n*n+1)/2);

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

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

  System.out.print((i+j)%2==0?"C":".");

}
```

```
System.out.println();

}



}

}
```

15

Chess Showmatch

1. **Problem statement:**

There is a showmatch on a chess competition.

To make it more fun it was decided to only present players that have a difference in their skill not greater than d.

Calculate how many ways of selecting a pair of players exists for a given array of players' skill levels.

2. Input Format:

The first line contains two integers n and d - the number of players and the maximum skill difference.

The second line contains n integers not greater than 10^9, representing the players' skill level.

3. Output Format:

Output the number of ways to select pairs of players whose skill difference is not greater than d. Note that pairs (1,5) and (5,1) are different.

4. Constraints:

$1 \le n \le 1000$, $1 \le d \le 10^{\wedge}9$

5. Samples Input/Output:

	Input	Output
1	5 10 10 20 50 60 65	6
2	5 1 55 30 29 31 55	6

6. Test cases

	Input	Output
1	7 100 19 1694 261 162 1 234 513	8
2	8 42 37 53 74 187 568 22 5 65	20
3	10 4 11 6 76 49 28 20 57 152 5 32	4
4	10 2 11 6 76 49 28 20 57 152 5 32	2
5	12 10 11 6 76 49 28 20 57 152 5 32 5 133	20
6	9 100000 52416871 133991223 7701764 69178544 112004989 38614509 264699878 183957443 97037454	0
7	11 100000 8025501 3837608 6924265 13473144 24563120 10091128 11683119 14010426 2991821 178890 6899200	2
8	8 100000	4

	982834295 993958517 998946898 982781729 980401767 998997082 990515494 996870812	

7. Solution (Java)

```
import java.io.*;

import java.util.*;

public class SampleClass {

public static void main(String arg[]) throws IOException {

Scanner o1 = new Scanner(System.in);

int n, d;

n = o1.nextInt();

d = o1.nextInt();

int a[] = new int[n];

int i, j, c = 0;

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

a[i] = o1.nextInt();

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

for (j = 0; j < n; j++)

if (i != j && Math.abs(a[i] - a[j]) <= d) c++;

System.out.print(c);

}

}
```

16
Color Wars

1. **Problem statement:**

Let's look at the war of three colors: red, green and blue.
If two colors meet in a fight, one of them will win and survive, the other will lose and die. If some color has more units than the other, it will beat the other one. When color x wins the fight with color y, x changes its color to z.
Two same colors (x and x), won't fight each other.

The war ends when all the colors are the same.
You know the number of units for every color.
You need to find the minimum number of fights required for the war to end.

2. Input Format:

The first line - x, y and z — the number of units of red, green , blue

3. Output Format:

The minimum number of fights required to end the war

4. Constraints:

$0 \le x, y, z \le 231$; $x + y + z > 0$

5. Samples Input/Output:

	Input data	Output data
1	1 1 1	1
2	3 1 0	3

6. Test Cases:

	Input data	Output data
1	1 4 4	4
2	5 10 6	10
3		8

	6 8 10	
4	1 10 2	10
5	10 6 8	8
6	18 67 5	67
7	67 81 1	67

7. Source (Java):

```
import java.util.*;

public class Main {

public static void main(String [] args){

        Scanner in=new Scanner(System.in);

        long array[]=new long[3];

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

        array[i]=in.nextLong();

        }
```

```
        Arrays.sort(array);

        if(array[1]%2==array[0]%2){

            System.out.print(array[1]);

            return;

        }

        System.out.print(array[2]);

}

}
```

17

Cost Game

1. **Problem statement:**

Mira plays a game where he needs to win as many points as he can. His friend Maja doesn't believe him and often wants to check if the result is true.

This game is composed of two rounds happening at the same time and last for T minutes, starting from zero. (0 .. T-1). The first round has the initial cost of A, and every minute the cost is reduced by DA. The second round has the initial cost B, and every minute it is reduced by DB.

(after zero minute, the costs will be A-DA, and B-DB - non-negative)

You have to check if Mira is telling the truth about his winning points.

2. Input Format:

A single line : *X*, T, *A*, *B*, *DA*, *DB* — Mira's result,the duration, the initial costs and the reduced costs per minute.

3. Output Format:

Print "YES" if Mira is telling the truth, otherwise print "NO". (without quotes)

4. Constraints:

$0 \le X \le 600$; $1 \le T, A, B, DA, DB \le 300$

5. Samples Input/Output:

	Input data	Output data
1	30 5 20 20 3 5	YES
2	10 4 100 5 5 1	NO

6. Test Cases:

	Input data	Output data
1	0 7 30 50 3 4	YES
2	50 10 30 20 1 2	YES
3	40 1 40 5 11 2	YES
4	35 8 20 20 1 2	YES

5	46 7 18 6 3 1	NO
6	2 5 29 36 5 6	NO
7	17 10 10 20 1 2	YES

7. Source (Java):

```
import java.util.Scanner;



public class TwoProblems {

    public static void main(String asd[])throws Exception

    {

        Scanner in=new Scanner(System.in);

        int x=in.nextInt();

        int t=in.nextInt();

        int a=in.nextInt();int b=in.nextInt();

        int q=in.nextInt();int w=in.nextInt();int i,k;

       for(i=0;i<t;i++)for(k=0;k<t;k++)if(a-q*i+b-w*k==x||a-q*i==x||
b-w*k==x||x==0){System.out.print("YES");return;}

       System.out.println("NO");
```

```
    }

}
```

18
Divisibility

1. **Problem statement:**

An IT-Town company that produces videogames came up with a new way to stimulate their employees. When there is a new game release, users start buying it and the company keeps track of the sales numbers up to each transaction. Whenever the next number of sales is divisible by all the numbers from 2 to 10, all developers receive a small bonus.

Game designer Peter knows that the company is about to release a new game, in the development of which he participated. Based on the past experience, he predicts that within a month the game will be purchased by n people. Now Peter wants to determine how many times he will receive a bonus. Help him find that out.

2. Input Format:

The only line of input contains one integer n - the number of people predicted to buy the game.

3. Output Format:

Print out a single number - the number of integers between 1 and n, divisible by all the numbers from 2 to 10.

4. Constraints:

$1 \le n \le 10^{18}$

5. Samples Input/Output:

	Input data	Output data
1	3000	1

2	2520	1

6. Test Cases:

	Input data	Output data
1	2521	1
2	314159265	124666
3	718281828459045235	285032471610732
4	1000000000000000000	396825396825396
5	987654321234567890	391926317950225
6	3628800	1440
7	5040000000000000000	200000000000000

7. Sample solution (Java):

```
import java.util.Scanner; public class J { public static void main(String[] args) {
Scanner scanner = new Scanner(System.in); long nextLong = scanner.nextLong();
System.out.println(nextLong / 2520); } }
```

19
James and Blocks

1. **Problem statement:**

James has a row of blocks, each of which has an integer painted on it. James can swap two adjacent blocks. Given a finite amount of time, can James get a row in which the numbers of any two adjacent blocks are different?

Help James.

2. Input Format:

The first line contains an integer n - the number of blocks. The second line contains n integer numbers a_1, a_2, ..., a_n - the numbers of the blocks.

3. Output Format:

On a single line print «YES» (without quotes), if James could get the desired array, and «NO» (without the quotes) otherwise.

4. Constraints:

$1 \le n \le 100$

$1 \le a_i \le 1000$

5. Samples Input/Output:

	Input data	Output data
1	1 1	YES
2	3 1 1 2	YES

6. Test Cases:

	Input data	Output data
1	4 7 7 7 7	NO
2	6 727 539 896 668 36 896	YES
3	8 742 742 742 742 742 289 742 742	NO
4	9 730 351 806 806 806 630 85 757 967	YES

5	7 674 712 674 674 674 674 674	NO
6	8 742 742 742 742 742 289 742 742	NO
7	9 730 351 806 806 806 630 85 757 967	YES

7. Sample solution (Java):

```
import  java.util.*;

public  class   Main{

    public  static  void    main(String[]args){

        Scanner cin=new Scanner(System.in);

        int n=cin.nextInt(),m=3+n>>1,S[]= new int[1000];

        for(;0<n--;++S[cin.nextInt()-1]);

        while(1000>++n&&m>S[n]);

        System.out.print(999<n?"YES":"NO");

    }

}
```

20

Little Sisters

1. **Problem statement:**

Hikki has n little sisters and n is an even number. He bought n^2 boxes of lollipops. And each m box contains m lollipops.

Help Hikki give each sister n boxes of lollipops so all the sisters get the same amount of lollipops.

2. Input Format:

The single line contains one integer n - the number of sisters Hikki has.

3. Output Format:

Print n lines with n integers each representing which boxes the sisters get. All these numbers should be distinct and be between 1 and n^2. It is guaranteed that a valid solution exists.

4. Constraints:

n - is even, $2 \le n \le 100$

5. Samples Input/Output:

	Input	Output
1	2	1 4 2 3
2	4	1 16 2 15 3 14 4 13 5 12 6 11 7 10 8 9

6. Test cases

	Input	Output
1	6	1 36 2 35 3 34 4 33 5 32 6 31 7 30 8 29 9 28 10 27 11 26 12 25 13 24 14 23 15 22 16 21 17 20 18 19
2	8	1 64 2 63 3 62 4 61 5 60 6 59 7 58 8 57 9 56 10 55 11 54 12 53 13 52 14 51 15 50 16 49 17 48 18 47 19 46 20 45 21 44 22 43 23 42 24 41 25 40 26 39 27 38 28 37 29 36 30 35 31 34 32 33
3	10	1 100 2 99 3 98 4 97 5 96 6 95 7 94 8 93 9 92 10 91 11 90 12 89 13 88 14 87 15 86 16 85 17 84 18 83 19 82 20 81 21 80 22 79 23 78 24 77 25 76 26 75 27 74 28 73 29 72 30 71 31 70 32 69 33 68 34 67 35 66 36 65 37 64 38 63 39 62 40 61 41 60 42 59 43 58 44 57 45 56 46 55 47 54 48 53 49 52 50 51
4	12	1 144 2 143 3 142 4 141 5 140 6 139 7 138 8 137 9 136 10 135 11 134 12 133 13 132 14 131 15 130 16 129 17 128 18 127 19 126 20 125 21 124 22 123 23 122 24 121 25 120 26 119 27 118 28 117 29 116 30 115 31 114 32 113 33 112 34 111 35 110 36 109

		37 108 38 107 39 106 40 105 41 104 42 103 43 102 44 101 45 100 46 99 47 98 48 97 49 96 50 95 51 94 52 93 53 92 54 91 55 90 56 89 57 88 58 87 59 86 60 85 61 84 62 83 63 82 64 81 65 80 66 79 67 78 68 77 69 76 70 75 71 74 72 73
5	14	1 196 2 195 3 194 4 193 5 192 6 191 7 190 8 189 9 188 10 187 11 186 12 185 13 184 14 183 15 182 16 181 17 180 18 179 19 178 20 177 21 176 22 175 23 174 24 173 25 172 26 171 27 170 28 169 29 168 30 167 31 166 32 165 33 164 34 163 35 162 36 161 37 160 38 159 39 158 40 157 41 156 42 155 43 154 44 153 45 152 46 151 47 150 48 149 49 148 50 147 51 146 52 145 53 144 54 143 55 142 56 141 57 140 58 139 59 138 60 137 61 136 62 135 63 134 64 133 65 132 66 131 67 130 68 129 69 128 70 127 71 126 72 125 73 124 74 123 75 122 76 121 77 120 78 119 79 118 80 117 81 116 82 115 83 114 84 113 85 112 86 111 87 110 88 109 89 108 90 107 91 106 92 105 93 104 94 103 95 102 96 101 97 100 98 99
6	16	1 256 2 255 3 254 4 253 5 252 6 251 7 250 8 249 9 248 10 247 11 246 12 245 13 244 14 243 15 242 16 241 17 240 18 239 19 238 20 237 21 236 22 235 23 234 24 233 25 232 26 231 27 230 28 229 29 228 30 227 31 226 32 225 33 224 34 223 35 222 36 221 37 220 38 219 39 218 40 217 41 216 42 215 43 214 44 213 45 212 46 211 47 210 48 209

		49 208 50 207 51 206 52 205 53 204 54 203 55 202 56 201 57 200 58 199 59 198 60 197 61 196 62 195 63 194 64 193 65 192 66 191 67 190 68 189 69 188 70 187 71 186 72 185 73 184 74 183 75 182 76 181 77 180 78 179 79 178 80 177 81 176 82 175 83 174 84 173 85 172 86 171 87 170 88 169 89 168 90 167 91 166 92 165 93 164 94 163 95 162 96 161 97 160 98 159 99 158 100 157 101 156 102 155 103 154 104 153 105 152 106 151 107 150 108 149 109 148 110 147 111 146 112 145 113 144 114 143 115 142 116 141 117 140 118 139 119 138 120 137 121 136 122 135 123 134 124 133 125 132 126 131 127 130 128 129
7	18	1 324 2 323 3 322 4 321 5 320 6 319 7 318 8 317 9 316 10 315 11 314 12 313 13 312 14 311 15 310 16 309 17 308 18 307 19 306 20 305 21 304 22 303 23 302 24 301 25 300 26 299 27 298 28 297 29 296 30 295 31 294 32 293 33 292 34 291 35 290 36 289 37 288 38 287 39 286 40 285 41 284 42 283 43 282 44 281 45 280 46 279 47 278 48 277 49 276 50 275 51 274 52 273 53 272 54 271 55 270 56 269 57 268 58 267 59 266 60 265 61 264 62 263 63 262 64 261 65 260 66 259 67 258 68 257 69 256 70 255 71

		254 72 253 73 252 74 251 75 250 76 249 77 248 78 247 79 246 80 245 81 244 82 243 83 242 84 241 85 240 86 239 87 238 88 237 89 236 90 235 91 234 92 233 93 232 94 231 95 230 96 229 97 228 98 227 99 226 100 225 101 224 102 223 103 222 104 221 105 220 106 219 107 218 108 217 109 216 110 215 111 214 112 213 113 212 114 211 115 210 116 209 117 208 118 207 119 206 120 205 121 204 122 203 123 202 124 201 125 200 126 199 127 198 128 197 129 196 130 195 131 194 132 193 133 192 134 191 135 190 136 189 137 188 138 187 139 186 140 185 141 184 142 183 143 182 144 181 145 180 146 179 147 178 148 177 149 176 150 175 151 174 152 173 153 172 154 171 155 170 156 169 157 168 158 167 159 166 160 165 161 164 162 163

7. Solution (Java)

```
import java.util.*;

public class a334 {

public static void main(String ar[]) {

Scanner ob = new Scanner(System.in);

int n = ob.nextInt();

int x = 1, y = n * n;

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

for (int j = 0; j < n / 2; j++) {

System.out.print(x + " " + y + " ");

x += 1;
```

```
y -= 1;
}
System.out.println();
}
}
}
```

21
Michael's Matrix

1. **Problem statement:**

Michael loves matrixes N x N. Michael has a special love for matrixes in which the sum of the elements in each row and each column is equal to K.

Help him find at least one such matrix.

2. Input Format:

N and K

3. Output Format:

A N x N matrix with all elements not greater than 1000 and not less than -1000.

4. Constraints:

$1 \le N \le 100, 1 \le K \le 1000$

5. Samples Input/Output:

	Input data	Output data
1	2 4	4 0 0 4
2		

	4 7	7 0 0 0 0 7 0 0 0 0 7 0 0 0 0 7

6. Test Cases:

	Input data	Output data
1	1 8	8
2	9 3	3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3
3	11 547	547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547 0 0

		0 0 0 0 0 0 0 0 0 547 0 0 0 0 0 0 0 0 0 0 0 547
4	2 7	7 0 0 7
5	4 3	3 0 0 0 0 3 0 0 0 0 3 0 0 0 0 3
6	3 2	2 0 0 0 2 0 0 0 2
7	1 1	1

7. Sample solution (Java):

```
import java.util.*;

public class Main{

public static void main(String... args){

Scanner sc = new Scanner(System.in);

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

```
int s = sc.nextInt();

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

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

    if (i==j) {System.out.print(s+" ");} else {System.out.print("0
");}

  }

  System.out.println();

}




}

}
```

22
My Class

1. **Problem statement:**

In my class there are n boys and m girls. We want to distribute them so that they would stand in a line, and the variations of boys and girls should be as great as possible. The positions are enumerated from left to right 1..n+m, and the number of integers is such that a position with the nearby right child will have different genders (for example i - boy, i+1 - girl) and must be as large as possible. You need to help with forming the line.

2. Input Format:

The first line - *n* and *m*, separated by a space.

3. Output Format:

Print a line of $n + m$ children without spaces: print the i-th position as "B", if it will be a boy, else print G (without quotes). Note that the number of "B"s should be equal to n, and the number of "G"s - to m. If multiple solutions exist, print out any of them.

4. Constraints:

$1 \le n, m \le 100$

5. Samples Input/Output:

	Input data	Output data
1	3 3	GBGBGB
2	4 2	BGBGBB

6. Test Cases:

	Input data	Output data
1	5 5	GBGBGBGBGB
2	6 4	BGBGBGBGBB
3	76 48	BGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBG BGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBG BGBGBGBGBGBGBGBGBGBGBGBGBBBBBBBBBBBB BBBBBBBBBBBBBBB

4	1 98	GBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
5	1 34	GBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG G Answer
6	1 4	GBGGG
7	18 94	GBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGBGB GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG GGGGGGGG

7. Source (Java):

```
import java.util.*;

import java.io.*;

public class Main{

public static void main(String... args) throws IOException{

Scanner sc = new Scanner(new File("input.txt"));

int b = sc.nextInt();

int g = sc.nextInt();

StringBuilder r = new StringBuilder();
```

```
if (b>g){

  r.append('B');

  b--;

}

while (b>0 || g>0){

  if (g>0){ r.append('G'); g--;}

  if (b>0){ r.append('B'); b--;}

}

PrintWriter pw = new PrintWriter("output.txt");

pw.println(r);

pw.flush();

pw.close();

}

}
```

23
Ocean Sponge

1. **Problem statement:**

One sponge living in a pineapple under the sea has found a stick. That stick was of an integer lengh n. The sponge wanted to find how many ways there were to split the stick into 4 little sticks so that the little sticks could form a rectangle but couldn't form a square. For some reason the Sponge hates squares. All little sticks should have an integer length and their combined length should be n.

2. Input Format:

The single line contains one integer n

3. Output Format:

Print out the number of ways to split the stick into 4 little sticks so that the little sticks could form a rectangle but couldn't form a square.

4. Constraints:

$1 \le n \le 2 \cdot 10^9$

5. Samples Input/Output:

	Input	Output
1	6	1
2	20	4

6. Test cases

	Input	Output
1	1	0
2	4	0
3	24	5
4	1995886626	498971656
5	1996193634	499048408
6	176409698	44102424
7	1829551191	0
8	71207034	17801758

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

if (n % 2 != 0) {

System.out.println(0);

return;

}

System.out.println((n - 2) / 4);

}

}
```

24

Relatives and a Cake

1. **Problem statement:**

The Fedorov family is preparing to celebrate the birthday of Thomas, this family prepared him a very peculiar cake.

The cake is a n × n square consisting of identical squares with the side 1. Each square is either empty or contains chocolate. Family members baked the cake and put chocolate on it. They think that Thomas' joy will be equal to the number of pairs of cells with chocolates arranged in a single row or in one column. Now they are trying to figure out what the value for this cake is.

Note that each pair cannot be counted more than once, since two different cells may not both be arranged in one row and one column.

2. Input Format:

The first line of input contains a single integer n - the length of the sides of the cake.

The following n lines contain n characters describing the cake itself. Empty cells are indicated by ".", And cells containing chocolate are marked with the symbol 'C'.

3. Output Format:

Output what the joy of Thomas will equal when he sees the cake, i.e. the number of cells with chocolates arranged in a single row or in one column.

4. Constraints:

$1 \le n \le 100$

5. Samples Input/Output:

	Input data	Output data
1	3 .CC C.. C.C	4
2	4 CC.. C..C .CC. .CC.	9

We number the rows from top to bottom and the columns from left to right. Then in one row there are:

(1, 2) and (1, 3)

(3, 1) and (3, 3)

In one column there are the following cells with chocolates:

(2, 1) and (3, 1)

(1, 3) and (3, 3)

6. Test Cases:

	Input data	Output data
1	5 .CCCC CCCCC .CCC. CC... .CC.C	46
2	6 C.CC.C ..C..C C..C.C .CCC.C CC.... CC....	39
3	7 .CC..CC CC.C..C C.C..C. C...C.C CCC.CCC .CC...C .C.CCC.	84
4	8 ..C....C C.CCC.CC .C..C.CC CC..... C..C..CC C.C...C. C.C..C.. C...C.C.	80
5	9 .C...CCCC C.CCCC... ....C..CC .CC.CCC.. .C.C..CC. C...C.CCC CCC.C...C CCCC....C ..C..C..C	144

6	10 ..C..C.C.. ..CC..C.CC .C.C...C.C ..C.CC..CC ....C..C.C ...C..C..C CC.CC....C ..CCCC.C.C ..CC.CCC.. CCCC..C.CC	190
7	11 C.CC...C.CC CC.C....C.C .....C..CCC ....C.CC.CC C..C..CC... C...C...C.. CC..CCC.C.C ..C.CC.C..C C...C.C..CC .C.C..CC..C .C.C.CC.C..	228

7. Sample solution (Java):

```
import java.util.Scanner; public class Sample{ public static Scanner sb; public static void main(String[] args){ sb=new Scanner(System.in); int n=sb.nextInt();int i,j,count,sum=0;String d; char a[][]=new char[n][n]; for(i=0;i<n;i++){ count=0; d=sb.next(); for(j=0;j<n;j++){ a[i][j]=d.charAt(j); if(a[i][j]=='C') count++; } sum+=(count*(count-1))/2; } for(j=0;j<n;j++){ count=0; for(i=0;i<n;i++){ if(a[i][j]=='C') count++; } sum+=(count*(count-1))/2; } System.out.println(sum); } }
```

25

Shuffle game

1. **Problem statement:**

A monkey puts a little red ball under one of three coconut shells. Then the monkey shuffles the shells by swapping them three times with each other.

A panda tries to guess where the ball will be after the three shuffles, but fails a lot and has asked for your help in the panda language.

2. Input Format:

The first line contains an integer k - the index of the coconut shell that is hiding the ball before shuffling.

Next three lines describe each shuffle movement with two integers - indices of the swapped coconut shells.

3. Output Format:

Output a single integer - the index of the she;; under which the ball will be after the shuffle.

4. Constraints:

$1 \leq k \leq 3$

5. Samples Input/Output:

	Input	Output
1	1 1 2 2 1 2 1	2
2	1 2 1 3 1 1 3	2

6. Test cases

	Input	Output
1	3 3 1 2 1	1

	1 2	
2	1 1 3 1 2 2 3	2
3	3 3 2 3 1 3 1	2
4	3 3 1 2 3 3 2	1
5	2 1 3 1 2 2 1	2
6	1 1 3 3 2 1 2	1
7	1 1 3 1 3 2 3	1
8	2 1 2 2 3	2

	2 1	

7. Solution (Java)

```
import java.util.Scanner;

import java.io.File;

import java.io.PrintWriter;

import java.io.FileWriter;

public class SampleClass {

public static void main(String[] args) throws Exception {

Scanner yo = new Scanner(System.in);

int n = yo.nextInt();

int[] cups = new int[3];

cups[n - 1] = 1;

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

int a = yo.nextInt();

int b = yo.nextInt();

int temp = cups[a - 1];

cups[a - 1] = cups[b - 1];

cups[b - 1] = temp;

}

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

if (cups[i] == 1) {

System.out.println(i + 1);

take.close();

break;

}

}
```

}

}

26

Tennis Championship

1. **Problem statement:**

There is a tennis tournament in which n people participate. The participants play according to the Olympic system where winners advance in the tournament and losers are eliminated.

The tournament grid is made up as follows (m is the number of participants of the current round):

k is found, equal to the maximum power of 2 that is $k \leq m$,

k participants compete in the current round, and half of them advance to the next round, the remaining m - k participants pass to the next round without competingt,

when there is only one participant left, the tournament is concluded.

For each match, b bottles of water are required for each participant and one bottle of for the judge. In addition, each player is given p towels for the entire tournament.

Determine the number of bottles and towels needed for the tournament.

Please note that this is tennis tournament, so two players compete (one of them wins, the other loses) every game.

2. Input Format:

The only line contains three integers n, b, p - the number of participants in the tournament, and the parameters described in the problem statement.

3. Output Format:

Enter two integers x and y - the number of bottles and towels needed for the tournament.

4. Constraints:

$1 \le n, b, p \le 500$

5. Samples Input/Output:

	Input data	Output data
1	5 2 3	20 15
2	8 2 4	35 32

In the first example will be three rounds:

In the first round there will be two matches, for each 5 bottles of water are required (two bottles of players and one judge)

In the second round there will be only one match, so we need another 5 bottles of water,

In the third round there will also be a match, so we need another 5 bottles of water.

Thus, all we need is 20 bottles of water.

In the second example, none of the participants will advance to any round directly.

6. Test Cases:

	Input data	Output data
1	10 1 500	27 5000
2	20 500 1	19019 20
3	100 123 99	24453 9900
4	500 1 1	1497 500
5	500 500 500	499499 250000
6	1 2 3	0 3
7	1 2 133	0 133

7. **Sample solution (Java):**

```
import java.io.BufferedReader; import java.io.IOException; import
java.io.InputStreamReader; import java.util.StringTokenizer; public class A { public
static void main(String[] args) throws IOException { BufferedReader br = new
BufferedReader(new InputStreamReader(System.in)); StringTokenizer st = new
StringTokenizer(br.readLine()); int n = Integer.parseInt(st.nextToken()); int b =
Integer.parseInt(st.nextToken()); int p = Integer.parseInt(st.nextToken()); long ret
= 0; int m = n; while (m > 1) { int pow = 2; while (pow < m) { pow *= 2; } ret +=
pow * b + (pow / 2); m -= (pow / 2); } System.out.println(ret + " " + p * n); } }
```

27

The game

1. **Problem statement:**

There is a legend in the university of IT-Town. To a student who did not answer a single question at the exam on the game theory, the teacher gives one last chance. The student has to play a game with the teacher.

The game is played on a square checkered field n × n. Initially, all the cells are empty. Each turn, a player selects an empty cell that has no common side with painted cells, and paints it. Touching painted cells with corners is allowed. On the next turn, so does the second player, after the first, and so on. The player who can't make a move in their turn loses.

The teacher has already chosen a field of size n, and provided student with the choice to have the first or the second turn. What should the student choose to win the game? We assume that both players play optimally.

2. Input Format:

The only line of input contains one integer n - the field size.

3. Output Format:

Print out number 1 if both players play optimally and the player who takes the first turn wins, otherwise print 2.

4. Constraints:

$1 \le n \le 10^{18}$

5. Samples Input/Output:

	Input data	Output data
1	1	1
2	2	2

6. Test Cases:

	Input data	Output data
1	3	1
2	4	2
3	5	1
4	6	2
5	1000000000000000000	2
6	999999999999999999	1
7	321392715309062180	2

7. Sample solution (Java):

```
import java.util.*;

import java.math.*;

public class Main

{

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

```
Scanner s = new Scanner(System.in);


long d=s.nextLong();

if(d%2==0)

System.out.println(2);

else

System.out.println(1);

}}
```

28

Timur the Rabbit

1. **Problem statement:**

Rabbit Timur is playing with numbers.

He has N integers: x1, x2, ..., xn. He can choose any I and J, if xi > xj, and do the following operation xi = xi - xj.

Find the minimum sum of all numbers after applying this operations any number of times.

2. Input Format:

The first line - N.

The next N lines - x_1, x_2, ..., x_n.

3. Output Format:

The minimum sum.

4. Constraints:

$2 \le N \le 100$

$1 \le x_i \le 100$

5. Samples Input/Output:

	Input data	Output data
1	2 1 2	2
2	3 2 4 6	6

6. Test Cases:

	Input data	Output data
1	2 12 18	12
2	5 45 12 27 30 18	15
3	2 1 1	2
4	2 100 100	200
5	2 87 58	58
6	10 60 12 96 48 60 24 60 36 60 60	120
7	20	1020

	51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51	

7. Sample solution (Java):

```
import java.util.*;

public class aa {

public static void main(String[] args)

{

    Scanner input = new Scanner(System.in);

    int n = input.nextInt();

    int g = input.nextInt();

    for(int i = 0; i<n-1; i++) g = gcd(g, input.nextInt());

    System.out.println(g*n);

}

static int gcd(int a, int b)

{

    return b == 0 ? a : gcd(b, a%b);

}

}
```

29
Trees on a Road

1. **Problem statement:**

Let's look at a road with n trees. Nadu wants to know how many ways he has to choose three trees so that the distance between two farthest trees is less than a given d.

Note that the order of the chosen threes doesn't matter.

2. Input Format:

The first line - *n* and *d*.

The next line - *n* integers *t_{1}, t_{2}, ..., t{n} - the trees' positions on the road*

3. Output Format:

The number of groups of three trees, so that the distance between two farthest points is less than the given *d*.

4. Constraints:

$1 \le n \le 105$; $1 \le d \le 109$

5. Samples Input/Output:

	Input data	Output data
1	4 3 1 2 3 4	4
2	4 2 -3 -2 -1 0	2

6. Test Cases:

	Input data	Output data
1	5 19 1 10 20 30 50	1

2	10 5 31 36 43 47 48 50 56 69 71 86	2
3	10 50 1 4 20 27 65 79 82 83 99 100	25
4	10 90 24 27 40 41 61 69 73 87 95 97	120
5	3 1000000000 -5 -1 1	1
6	2 1000000000 -14348867 1760823	0
7	1 14751211 847188590	0

7. Source (Java):

```
import java.util.*;

public class A{

    public static void main(String [] args){

        Scanner s = new Scanner(System.in);
```

```
        final int n = s.nextInt();
        final long d = s.nextLong();
        long [] x = new long[n];
        for (int i = 0; i < n; ++i){
            x[i] = s.nextLong();
        }
        long res = 0;
        for (int i = 0; i < n; ++i){
            long pos = Arrays.binarySearch(x, i, n, x[i] + d + 1);
            if (pos < 0){
                pos = -pos - 1;
            }
            if (pos - i >= 3){
                res += ((pos - i - 1)*(pos - i - 2))/2;
            }
        }
        System.out.println(res);
    }
}
```

30
Triangle Construction

1. **Problem statement:**

You are given three numbers a,b and c. You may increase any of these numbers by the total of l.

Calculate the number of ways to create a triangle with side lengths of a,b and c, that has a positive area.

The given and the increased numbers must be integers.

In the first test case you can either not increase any number or increase any two numbers by 1.

2. Input Format:

The first line contains four integers a, b, c, l .

3. Output Format:

Output the number of ways to increase a,b and c so you can build a triangle with a positive area from them.

4. Constraints:

$1 \le a, b, c \le 3 \cdot 10^5, 0 \le l \le 3 \cdot 10^5$

5. Samples Input/Output:

	Input	Output
1	1 1 1 2	4
2	1 2 3 1	2

6. Test cases

	Input	Output
1	10 2 1 7	0
2	1 2 1 5	20

3	10 15 17 10	281
4	100000 300000 100000 100100	255131325
5	100000 300000 200001 0	1
6	1 1 1 300000	1125022500250001
7	3 1 29 1	0
8	300000 300000 300000 300000	4500090000549998

7. Solution (Java)

```
import java.util.*;

import java.io.*;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

long a = sc.nextLong(), b = sc.nextLong(), c = sc.nextLong(), l = sc.nextLong();

long total = (l + 1) * (l + 2) * (l + 3);

total /= 6 L;

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

long k = Math.min(a - b - c + i, l - i);

if (k >= 0)

total -= ((k + 2) * (k + 1)) / 2;

k = Math.min(-a + b - c + i, l - i);

if (k >= 0)

total -= ((k + 2) * (k + 1)) / 2;
```

```
k = Math.min(-a - b + c + i, l - i);
if (k >= 0)
total -= ((k + 2) * (k + 1)) / 2;
}
System.out.println(total);
}
}
```

31
99 Luftballons

1. **Problem statement:**

Finn has 99 helium balloons of three colors, R balloons of red color, G balloons of green color and 99-R-G balloons of blue color. Finn wants to attach these balloons to his long fence, so that the balloons will form a line. However, Finn wants to use only two different colors and wants all adjacent balloons in line to have different color.

What is the maximum length of such line made of the balloons that Finn has?

2. Input Format:

The first line contains two positive integer numbers R, G.

3. Output Format:

Output the maximum length of such line made of the balloons that Finn has.

4. Constraints:

$0 \le R, G \le 99, 0 \le 99 - R - G$

5. Samples Input/Output:

	Input	Output
1	0 0	1
2	1 1	3

6. Test cases

	Input	Output
1	33 33	66
2	49 0	99
3	67 8	49
4	10 10	21
5	78 1	41
6	13 37	75
7	99 0	1
8	48 8	87

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int R = in .nextInt();

int G = in .nextInt();

int[] a = {R, G, 99 - R - G};

Arrays.sort(a);

int max = a[2];

int min = a[1];

int count = min * 2;

if (max > min)

count++;

System.out.print(count);
```

}

}

32
Acoustic Wavelengths

1. **Problem statement:**

Calculate the wavelengths of the first octave's notes for the given sound speed.

The frequencies are: 261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88 (Hz).

2. Input Format:

The sound speed, m/s.

3. Output Format:

Seven floating point values, meters.

4. Constraints:

-

5. Samples Input/Output:

	Input	Output
1	335	1.2804342009708367 1.1407750459715316 1.0162909929314687 0.9592532142141281 0.8545918367346939 0.7613636363636364 0.6783024216408844
2	0	0.0 0.0 0.0 0.0 0.0 0.0 0.0

6. Test cases

	Input	Output
1	1000	3.822191644689065 3.4052986446911393 3.0337044565118467 2.863442430489935 2.5510204081632653 2.272727272727273 2.0247833481817445

2	1000.0	3.822191644689065 3.4052986446911393 3.0337044565118467 2.863442430489935 2.5510204081632653 2.272727272727273 2.0247833481817445
3	335.0	1.2804342009708367 1.1407750459715316 1.0162909929314687 0.9592532142141281 0.8545918367346939 0.7613636363636364 0.6783024216408844
4	336	1.2842563926155257 1.1441803446162226 1.0193246973879804 0.9621166566446181 0.8571428571428571 0.7636363636363637 0.6803272049890662
5	337	1.288078584260215 1.147585643260914 1.0223584018444922 0.9649800990751081 0.8596938775510204 0.7659090909090909 0.682351988337248
6	338	1.291900775904904 1.150990941905605 1.025392106301004 0.9678435415055979 0.8622448979591837 0.7681818181818182 0.6843767716854297
7	339	1.295722967549593 1.154396240550296 1.028425810757516 0.9707069839360879 0.8647959183673469 0.7704545454545455 0.6864015550336114

7. Solution (Java):

```
private static final double[] freqs = {261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88};

public static void main(String[] args) {

   double speed = new Scanner(System.in).useLocale(Locale.US).nextDouble();

for (double freq : freqs) {

      System.out.print(speed/freq + " ");

 }
```

}

33
Adjustment

1. **Problem statement:**

Given a number, you should make another one from it.

Allowed actions:

- divide the number by 2, cut the fractional part (e. g. 3/2 = 1)
- multiply the number by 2

2. Input Format:

The first number n — the amount of numeric pairs.

The next n pairs of numbers — the source and target numbers.

3. Output Format:

The amount of actions required to transform the source numbers to target ones.

4. Constraints:

-

5. Samples Input/Output:

	Input	Output
1	3 1 4 4 512 9 16	2 7 3
2	3 3 8192 25 4	14 6 3

	1 8	

6. Test cases

	Input	Output
1	2 1 8 8 1	3 3
2	2 93 256 93 2	12 5
3	2 16777216 8 32768 2	21 14
4	2 1 2 2 1	1 1
5	2 768 128 777 128	16 16
6	1 1 1	0

7	1 768 768	0

7. Solution (Java):

```
public static void main(String args[]) {
   Scanner sc = new Scanner(System.in);

for (int count = sc.nextInt(); count > 0; count--) {
      int src = sc.nextInt(), dest = sc.nextInt(), opCount = 0;

while (src != dest) {
         if (dest % src == 0) {
            src *= 2;
 opCount++;
 } else {
            if (src % 2 == 0) {
               src /= 2;
 opCount++;
 } else {
               src = (src - 1) / 2;
 opCount++;
 }
         }
      }
      System.out.println(opCount);
 }
}
```

34
Android Problem

1. **Problem statement:**

Your Android application is still bad at displaying strings of different lengths, so you decided to add the necessary amount of symbols "_" to the start of the shorter strings, so that all strings have the same length.

2. Input Format:

The first line contains two strings s1 and s2. One string may be shorter than the other.

3. Output Format:

Output the shorter string appended with the necessary amount of symbols "_". If the strings are of the same length, output the first one.

4. Constraints:

$1 \le |s1|, |s2| \le 1000$

5. Samples Input/Output:

	Input	Output
1	aaaaaa a	_____a
2	android ios	_____ios

6. Test cases

	Input	Output
1	abcdef def	___def
2	abc def	abc
3	zzzzzzzzzzzzzzzzzzzzzzzzzzzzzz z	_____z
4	sdfsd sdfsfsfsfsfsfsdfsdfsdfsdf	_____sdfsd

5	q q	q
6	q qqqdqdqkdqkdqkdkqdkqdqkdkqdklkdafl,zmcvlkadsf,masdflkj	_____q
7	q1337222332323qqq kkkk	_____kkkk
8	jojo kokiko_kokiko_kokiko	_____jojo

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String s1 = in.next();

String s2 = in.next();

if(s1.length() == s2.length()){

System.out.println(s1);

return;

}

String s = s1.length() < s2.length() ? s1 : s2;

int difference = Math.abs(s1.length() - s2.length());

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

s = "_" + s;


System.out.println(s);

}

}
```

35
Apple Jake

1. **Problem statement:**

Jake stands in an apple tree grove. There are n apple trees and each apple tree has ai apples. Jake will pick one apple at a time from a tree that has the most apples. If there are multiple trees that have the most apples, Jake will choose the tallest one. Apple trees are sorted by height, and the first (index 0) is the tallest.

Jake is going to pick m apples, calculate which apple tree will be the last from which he picks an apple.

2. Input Format:

The first line contains two positive integer numbers: *n, m* - the number of trees, and number of apples Jake is going to pick. The second line contains n integers ai - the number of apples on the i-th tree

3. Output Format:

Output the index of the tree, from which Jake picks the last apple. Trees are numbered from 0 to n-1.

4. Constraints:

$1 \le n, m \le 1000$

$1 \le ai \le 1\ 000\ 000$

5. Samples Input/Output:

	Input	Output
1	2 3 1 5	1
2	6 10 1 2 3 4 5 6	5

6. Test cases

	Input	Output

1	6 1 1 1 1 1 1 1	0
2	10 50 1 2 31 24 1 2 1 2 1 22	9
3	9 14 8 8 8 2 2 2 1 1 1	1
4	20 100 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	9
5	24 188 8 8 8 10 4 2 6 43 8 9 10 11 12 30 13 14 15 40 16 17 34 18 11 13	10
6	22 600 311 100 4 54 84 339 10 11 12 20 23 12 112 23 15 80 16 17 60 18 100 60	15
7	22 67 84 81 82 71 23 6 72 37 72 3 7 1 23 71 2 3 7 1 8 3 2 11	2
8	31 300 84 3 5 3 4 5 8 9 17 8 93 49 1 8 2 3 4 71 12 34 1 2 3 7 41 92 34 79 81 4 7	25

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

int m = in .nextInt();

int[] apples = new int[n];

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

apples[i] = in .nextInt();
```

```
int max=0, pos=0;
while(m!=0)
{
max=0;
pos=0;
for(int i=0;i<n;i++)
{
if(max < apples[i])
{
max=apples[i];
pos=i;
}
}
apples[pos]--;
m--;
}
System.out.println(pos);
}
}
```

36
Archives

1. **Problem statement:**

You are extracting n 7z archives simultaneously. For each extraction process si - the speed of writing files on the hard drive in Mb/sec and ti - the time left in seconds - are known. Consider the hard drive speed to be the pinch-point of the process. So the maximum speed of file extraction would be the sum of all si. And the size of the i-th archive's content equals si * ti.

Calculate how many seconds will pass before all archives are extracted.

2. Input Format:

The first line contains one positive integer number: *n* - the number of archives. The second line contains n integers si - the speeds in Mb/sec of archive extraction.

The second line contains n integers ti - the time remaining.

3. Output Format:

Output the number of seconds that will pass before all archives are extracted.

4. Constraints:

$1 \le n \le 1000$

$1 \le si \le 1000$

$1 \le ti \le 1000$

5. Samples Input/Output:

	Input	Output
1	3 1 1 1 10 10 10	10.0
2	3 1 3 5 2 4 6	4.888888888888889

6. Test cases

	Input	Output
1	6 1 2 3 2 1 2 5 6 5 4 2 3	4.363636363636363
2	10 1 2 3 2 1 2 1 4 99 10	92.288

	100 100 100 2 3 3 3 5 100 100	
3	4 25 2 2 2 20 5 5 5	17.096774193548388
4	16 9 8 1 8 2 9 9 1 8 2 8 9 19 1 2 9 9 18 9 13 2 11 9 1 8 2 9 11 21 1 2 11	12.047619047619047
5	10 1 3 5 7 9 2 4 6 8 10 3 5 7 9 9 2 4 10 10 10	8.309090909090909
6	12 67 31 21 3 48 48 77 6 90 56 88 12 88 63 25 55 67 71 80 46 97 63 89 13	76.49908592321755
7	16 41 60 76 23 1 5 8 44 18 62 6 10 42 17 19 14 53 89 93 55 58 92 100 60 31 85 34 13 98 70 75 24	74.06502242152466
8	6 90 120 130 140 150 200 8 1 3 4 5 2	3.5421686746987953

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

int[] s = new int[n];
```

```
int[] t = new int[n];

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

s[i] = in .nextInt();

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

t[i] = in .nextInt();

long size = 0;

long speed = 0;

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

speed += s[i];

size += s[i] * t[i];

}

System.out.println((double)size / (double)speed);

}

}
```

37
Artmoney

1. **Problem statement:**

James has prepared a report on his little shop's profit. There are n different donuts in his shop. For each donut, the following information is known: the price of production, the retail price and the number of sold units.

Now James wants to find the most profitable donut, help him with that. The profit is calculated as (retail price - production price) * sold units.

2. Input Format:

The first line contains one positive integer number: n - the number of elements. The second line contains n integers prod$i$.

The third line contains n integers price$i$. The fourth line contains n integers sold$i$.

3. Output Format:

Output the index of the most profitable donut (number from 0 to n-1).

4. Constraints:

$1 \le n \le 1000$

$1 \le prod_i \le 1\ 000\ 000$

$1 \le price_i \le 1\ 000\ 000$

$1 \le sold_i \le 1\ 000\ 000$

5. Samples Input/Output:

	Input	Output
1	2 1 1 2 2 3 10	1
2	2 100 200 100 201 50 50	1

6. Test cases

	Input	Output
1	3 100 200 100 100 201 300 50 50 4	2
2	5 5 80 1000 10000 100000 10 100 1000 10020 100500	4

	1 30 5 7 9	
3	11 1 3 5 7 9 11 12 13 14 15 16 33 3 5 7 9 11 12 13 14 15 16 1 1 1 1 1 1 1 1 1 1 1	0
4	20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 22 33 44 55 66 77 88 99 100 111 1111 123 1337 1488 87 88 99 21 333 1 2 3 4 5 6 12 13 14 15 16 17 18 19 20 7 8 9 10 11	14
5	24 23 3 13 14 15 7 6 5 10 4 40 16 17 34 18 11 44 6 43 1 2 11 11 12 7 6 5 10 4 2 6 43 1 2 11 11 12 33 13 14 15 40 16 17 34 18 11 44 1 2 3 2 3 4 3 2 1 22 33 44 1 2 4 7 8 9 1 938 828 91928 838 8	21
6	20 1 3 2 4 79 83 705 98 1 370 59 18 3 7 4 5 0 1 9 45 89 12 37 9 18 23 70 91 28 37 12 98 3 7 76 6 12 389 2 17 37 9 89 12 18 23 76 6 12 389 2 12 98 3 7 17 70 91 28 37	17
7	20 14 15 16 17 18 6 7 8 9 19 20 1 2 3 4 5 10 11 22 33 10 20 30 40 50 60 70 80 90 100 101 102 103 104 105 106 107 108 109 100 100 200 300 200 100 200 300 200 100 200 300 200 100 200 300 200 100 200 300 200	14
8	29 5543 3 4776 5 8 91 17 8000 9 49123 1231 238 2342 324 4 7 12 34 1123 2312 13 7234 41 92 304 79 81 54 7000 5543 3 4776 5 8 93 17 8000 9 49123 1231 238 2342 324 4 7 12 34 1123 2312 3111 7234 41 92 304 79 81 14 7000	20

	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

7. Solution (Java)

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

int[] prod = new int[n];

int[] price = new int[n];

int[] sold = new int[n];

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

prod [i] = in .nextInt();

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

price [i] = in .nextInt();

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

sold[i] = in .nextInt();

int maxProfit = 0;

int best = 0;

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

int profit = (price[i] - prod[i]) * sold[i];

if(profit > maxProfit){

maxProfit = profit;

best = i;

}

}

System.out.println(best);

}
```

}

38
Bee Yxpress

1. **Problem statement:**

A group of ants want to reach the top of a tree. For that they decided to ride colored bees.

A colored bee arrives to the bottom of the tree every minute in the following order R - red, G - green, B - blue, and then again and again in the same order.

Each bee can carry 2 ants on its back. It takes 30 minutes for a bee to reach the top of the tree.

Ants are divided into three groups r,g и b, each group will only ride the bees of its color.

The first bee to arrive to the bottom of the tree is colored red. Calculate the time it takes the bees to transfer all ants to the top of the tree.

2. Input Format:

The input line contains three integers r, g and b.

3. Output Format:

Print out a single number — the time it takes the bees to transfer all ants to the top of the tree.

4. Constraints:

$0 \le r, g, b \le 100,\ r + g + b > 0$

5. Samples Input/Output:

	Input	Output
1	1 3 2	34
2	3 2 1	33

6. Test cases

	Input	Output

1	3 5 2	37
2	10 10 10	44
3	29 7 24	72
4	28 94 13	169
5	90 89 73	163
6	0 1 2	32
7	29 28 30	74

7. Solution (Java)

```
import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int r = in .nextInt();

int g = in .nextInt();

int b = in .nextInt();

int g1 = ((r + 1) / 2 - 1) * 3 + 30;

int g2 = ((g + 1) / 2 - 1) * 3 + 31;

int g3 = ((b + 1) / 2 - 1) * 3 + 32;

System.out.println(Math.max(g1, Math.max(g2, g3)));

}

}
```

39
Bijection

1. **Problem statement:**

You're given a mathematical function defined as two arrays x[n] and y[n], where y = f(x).

Define if the given function is a bijection, when each x value has a corresponding unique y value and all x values are unique too.

2. Input Format:

The first line contains one positive integer number: n - the number of elements. The second line contains n integers xi - the values of x.

The third line contains n integers yi - the values of y.

3. Output Format:

Output "YES" if the given function is a bijection, and "NO" otherwise.

4. Constraints:

$1 \le n \le 1000$

$1 \le xi \le 1\ 000\ 000$

$1 \le yi \le 1\ 000\ 000$

5. Samples Input/Output:

	Input	Output
1	3 1 2 3 1 2 3	YES
2	3 1 2 3 3 3 3	NO

6. Test cases

	Input	Output
1	6 1 8 12 99 4 11 1 2 8 9 3 1	NO

2	5 1 3 5 7 9 10 100 1000 10000 100000	YES
3	11 1 3 5 7 9 11 12 13 14 15 16 33 3 5 7 9 11 12 13 14 15 16	YES
4	20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 22 33 44 55 66 77 88 99 100 111 1111 123 1337 1488 87 88 99 21 333	NO
5	24 7 6 5 10 4 2 6 43 1 2 11 11 12 33 13 14 15 40 16 17 34 18 11 44 1 2 3 2 3 4 3 2 1 22 33 44 1 2 4 7 8 9 1 938 828 91928 838 8	NO
6	22 311 100 4 54 84 339 10 11 12 20 23 12 112 23 15 80 16 17 60 18 100 60 67 78 89 90 1 2 3 4 6 7 8 9 12 13 14 15 16 17 18 19 222 666	NO
7	20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 10 20 30 40 50 60 70 80 90 100 101 102 103 104 105 106 107 108 109 200000	YES
8	29 5543 3 4776 5 8 91 17 8000 9 49123 1231 238 2342 324 4 7 12 34 1123 2312 13 7234 41 92 304 79 81 54 7000 5543 3 4776 5 8 93 17 8000 9 49123 1231 238 2342 324 4 7 12 34 1123 2312 3111 7234 41 92 304 79 81 14 7000	YES

7. Solution (Java)

```
import java.util.*;

public class Main {
```

```
public static void main(String[] args) {
Scanner in = new Scanner(System.in);
int n = in .nextInt();
int[] x = new int[n];
int[] y = new int[n];
for (int i = 0; i < n; i++)
x[i] = in .nextInt();
for (int i = 0; i < n; i++)
y[i] = in .nextInt();
Set<Integer> set = new HashSet<Integer>();
boolean bijection = true;
for(int i = 0; i < n; i++){
if(set.contains(x[i])){
bijection = false;
break;
}
set.add(x[i]);
if(set.contains(-y[i])){
bijection = false;
break;
}
set.add(-y[i]);
}
System.out.println(bijection ? "YES" : "NO");
}
}
```
