HACKERRANK PROBLEM SOLVING

Alice and Bob each created one problem for HackerRank. A reviewer rates the two challenges, awarding points on a scale from 1 to 100 for three categories: problem clarity, originality, and difficulty.

The rating for Alice's challenge is the triplet a = (a[0], a[1], a[2]), and the rating for Bob's challenge is the triplet b = (b[0], b[1], b[2]).

The task is to find their comparison points by comparing a[0] with b[0], a[1] with b[1], and a[2] with b[2].

- If a[i] > b[i], then Alice is awarded 1 point.
- If a[i] < b[i], then Bob is awarded 1 point.
- If a[i] = b[i], then neither person receives a point.

Comparison points is the total points a person earned.

Given a and b, determine their respective comparison points.

Example

a = [1, 2, 3]

b = [3, 2, 1]

- For elements *0*, Bob is awarded a point because a[0] .
- For the equal elements a[1] and b[1], no points are earned.
- Finally, for elements 2, a[2] > b[2] so Alice receives a point.

The return array is [1, 1] with Alice's score first and Bob's second.

https://www.hackerrank.com/challenges/compare-the-triplets/problem

```
int* compareTriplets(int a_count, int* a, int b_count, int* b, int* result_count) {
    *result_count = 2;
    int *answer = malloc(*result_count * sizeof(int));
    int alice, bob = 0;
    for (int i = 0; i < 3; i++ ) {
        if (a[i] > b[i]) alice++;
        if (a[i] < b[i]) bob++;
    }
    answer[0] = alice;
    answer[1] = bob;
    return answer;
```

Given a square matrix, calculate the absolute difference between the sums of its diagonals.

For example, the square matrix arr is shown below:

```
1 2 3
4 5 6
9 8 9
```

The left-to-right diagonal = $1 + 5 + 9 = 15$. The right to left diagonal = $3 + 5 + 9 = 17$. Their absolute difference is $|15 - 17| = 2$.

Function description

Complete the *diagonalDifference* function in the editor below.

diagonalDifference takes the following parameter:

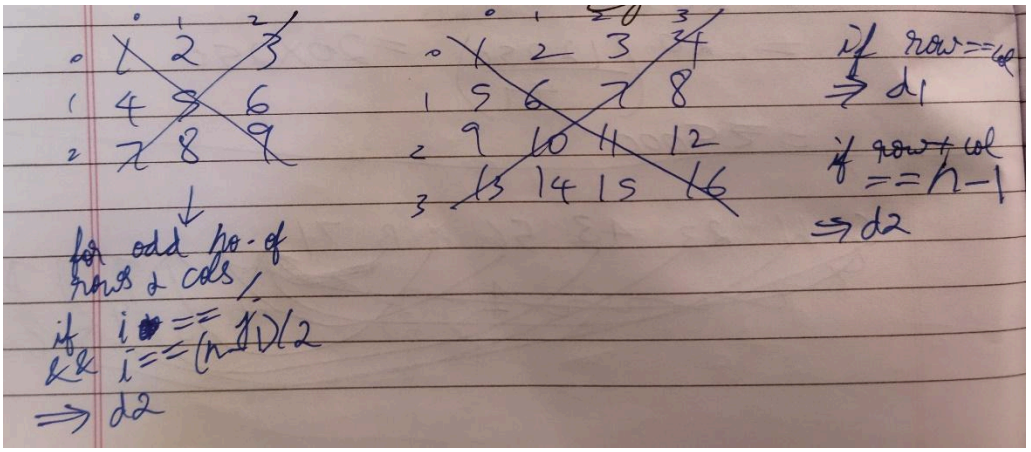
- int arr[n][m]: an array of integers

Return

- int: the absolute diagonal difference

https://www.hackerrank.com/challenges/diagonal-difference/problem?isFullScreen=false

```
int diagonalDifference(int arr_rows, int arr_columns, int** arr) {
    int d1=0, d2=0;
    int diff;

    for(int i=0;i<arr_rows;i++){
        for(int j=0;j<arr_columns;j++){
            if (i==j){
                d1+=arr[i][j];
            }
            else if (i+j==arr_rows-1){
                d2+=arr[i][j];
            }

            if (i==j && arr_rows%2!=0 && i==(arr_rows-1)/2){
                d2+=arr[i][j];
            }
        }
    }

    return abs(d1-d2);
```



HackerLand University has the following grading policy:

- Every student receives a $grade$ in the inclusive range from 0 to 100.
- Any $grade$ less than 40 is a failing grade.

Sam is a professor at the university and likes to round each student's $grade$ according to these rules:

- If the difference between the $grade$ and the next multiple of 5 is less than 3, round $grade$ up to the next multiple of 5.
- If the value of $grade$ is less than 38, no rounding occurs as the result will still be a failing grade.

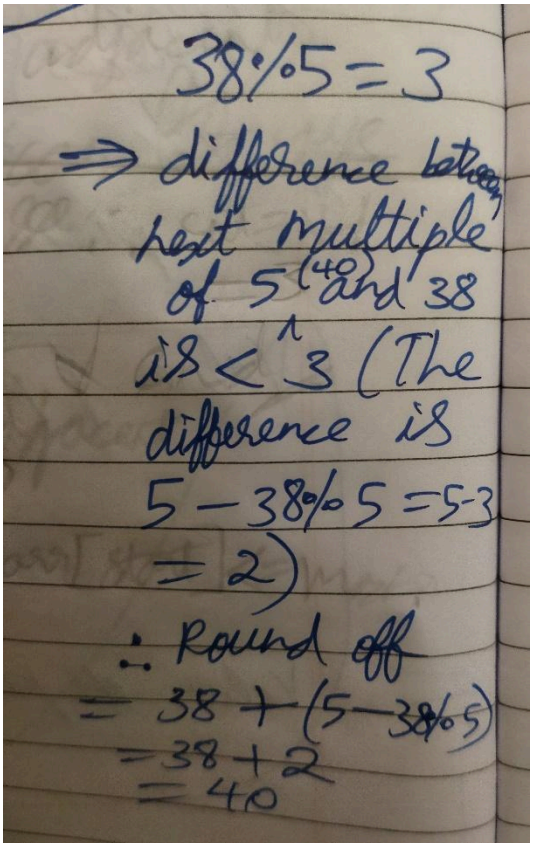
Examples

- $grade = 84$ round to 85 (85 - 84 is less than 3)
- $grade = 29$ do not round (result is less than 40)
- $grade = 57$ do not round (60 - 57 is 3 or higher)

Given the initial value of $grade$ for each of Sam's n students, write code to automate the rounding process.

https://www.hackerrank.com/challenges/grading/problem?isFullScreen=false

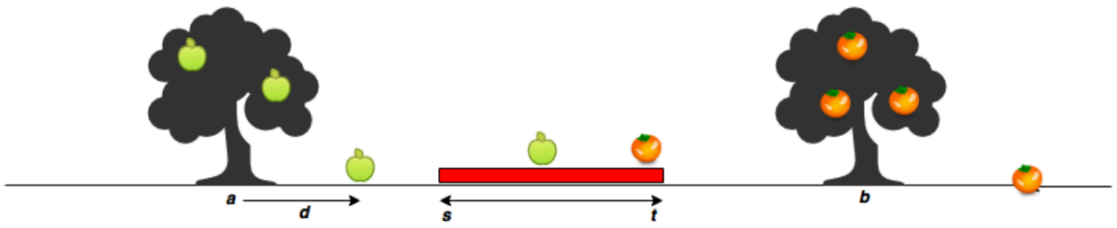
```
n = int(input().strip())
for a0 in range(n):
    x = int(input().strip())

    if x >= 38 and x % 5 > 2:
            x=x+(5-x%5)
    print(x)
```



Sam's house has an apple tree and an orange tree that yield an abundance of fruit. Using the information given below, determine the number of apples and oranges that land on Sam's house.

In the diagram below:

- The red region denotes the house, where s is the start point, and t is the endpoint. The apple tree is to the left of the house, and the orange tree is to its right.
- Assume the trees are located on a single point, where the apple tree is at point a, and the orange tree is at point b.
- When a fruit falls from its tree, it lands d units of distance from its tree of origin along the x-axis. *A negative value of d means the fruit fell d units to the tree's left, and a positive value of d means it falls d units to the tree's right. *

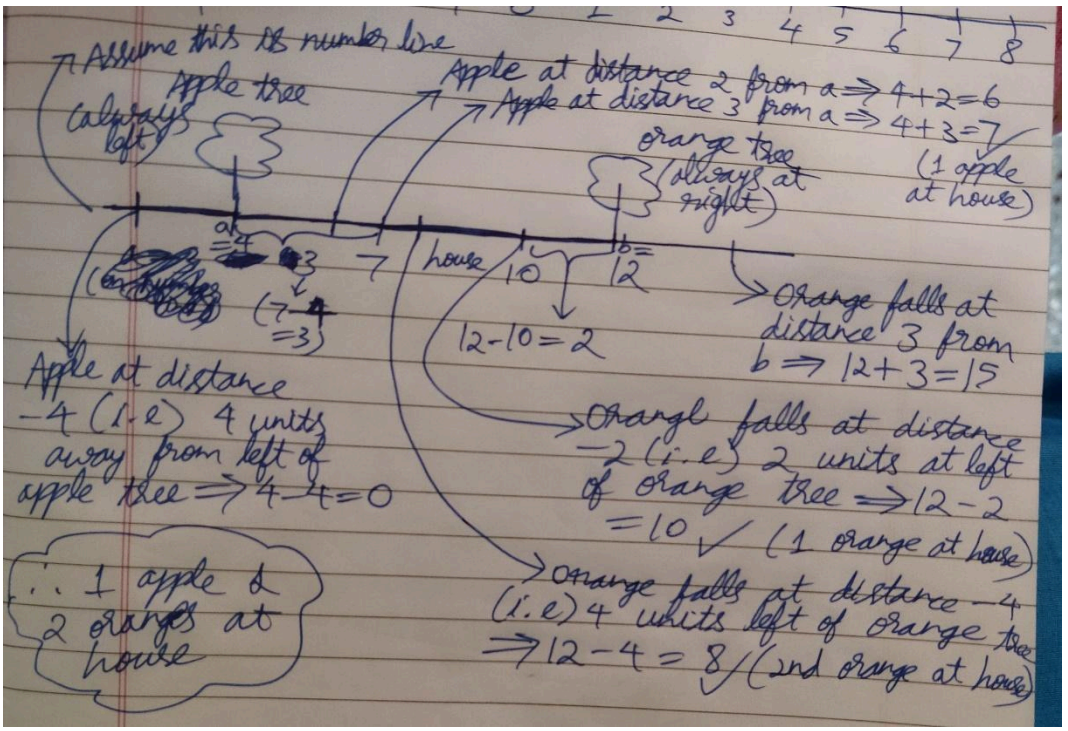


Given the value of d for m apples and n oranges, determine how many apples and oranges will fall on Sam's house (i.e., in the inclusive range $[s, t]$)?

For example, Sam's house is between $s = 7$ and $t = 10$. The apple tree is located at $a = 4$ and the orange at $b = 12$. There are $m = 3$ apples and $n = 3$ oranges. Apples are thrown $apples = [2, 3, -4]$ units distance from a, and $oranges = [3, -2, -4]$ units distance. Adding each apple distance to the position of the tree, they land at $[4 + 2, 4 + 3, 4 + -4] = [6, 7, 0]$. Oranges land at $[12 + 3, 12 + -2, 12 + -4] = [15, 10, 8]$. One apple and two oranges land in the inclusive range $7 - 10$ so we print

```
1
2
```

https://www.hackerrank.com/challenges/apple-and-orange/problem?isFullScreen=false




```
void countApplesAndOranges(int s, int t, int a, int b, int apples_count, int* apples, in
oranges_count, int* oranges) {
    int ac=0, oc=0;

    for(int i=0;i<apples_count;i++){
        int d=apples[i]+a;
        if(d>=s && d<=t){
            ac++;
        }
    }

    for(int i=0;i<oranges_count;i++){
    int d=oranges[i]+b;
    if(d>=s && d<=t){
        oc++;
    }
}
    printf("%d\n",ac);
    printf("%d",oc);

}
```

You are choreographing a circus show with various animals. For one act, you are given two kangaroos on a number line ready to jump in the positive direction (i.e, toward positive infinity).

- The first kangaroo starts at location $x1$ and moves at a rate of $v1$ meters per jump.
- The second kangaroo starts at location $x2$ and moves at a rate of $v2$ meters per jump.

You have to figure out a way to get both kangaroos at the same location at the same time as part of the show. If it is possible, return `YES`, otherwise return `NO`.

Example

$x1 = 2$

$v1 = 1$

$x2 = 1$

$v2 = 2$

After one jump, they are both at $x = 3$, ($x1 + v1 = 2 + 1$, $x2 + v2 = 1 + 2$), so the answer is `YES`.

https://www.hackerrank.com/challenges/kangaroo/problem?isFullScreen=false

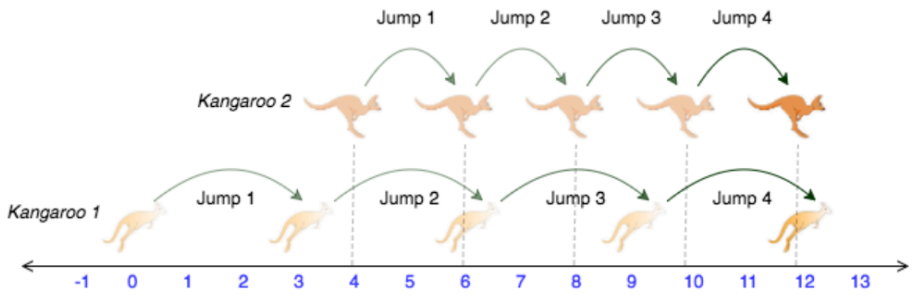
```
0 3 4 2
```

Sample Output 0

```
YES
```

Explanation 0

The two kangaroos jump through the following sequence of locations:



From the image, it is clear that the kangaroos meet at the same location (number 12 on the number line) after same number of jumps (4 jumps), and we print `YES`.

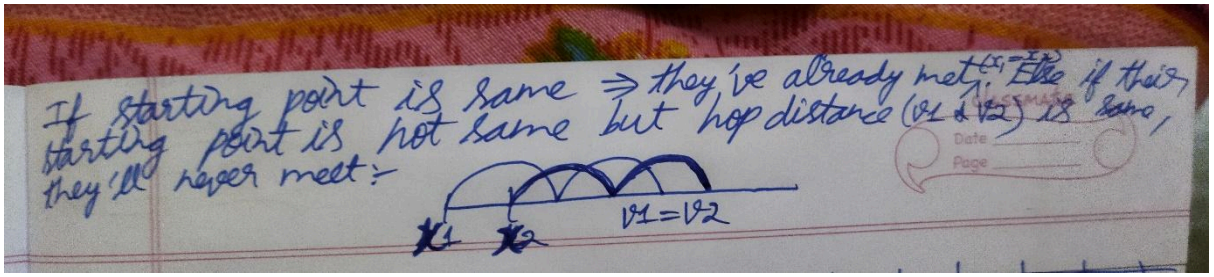
If $v_1 \leq v_2$, they will never meet.

It is required to check if a solution exists for the following equation:

$$x_1 + t * v_1 == x_2 + t * v_2$$

This is equivalent to checking if $(x_2 - x_1) \% (v_1 - v_2) == 0$.

```
x1,v1,x2,v2 = input().strip().split(' ')
x1,v1,x2,v2 = int(x1),int(v1),int(x2),int(v2)

if x1 == x2:
    print("YES")
elif v1 == v2:
    print("NO")
else:
    t = (x2-x1)/(v1-v2)
    if t%1 == 0 and t > 0:
        print("YES")
    else:
        print("NO")
```



Sure, let me explain the `if-else` conditions involving `t` in the provided code:

1. `if x1 == x2:`
    - If the starting positions of both cats are the same (`x1 == x2`), it means they are already at the same point. Therefore, they will meet, and the code prints `"YES"`.
2. `elif v1 == v2:`
    - If the velocities (jumping distances) of both cats are the same (`v1 == v2`), it means they will always maintain the same relative distance between them. In this case, if their starting positions are different (`x1 != x2`), they will never meet, and the code prints `"NO"`.
3. `else:`
    - If both the starting positions and velocities are different, the code calculates the time `t` when they will meet using the formula:

```
t = (x2 - x1) / (v1 - v2)
```

This formula is derived from the following equations:

```
x1 + v1 * t = x2 + v2 * t  (The positions of both cats are equal when they meet)
t = (x2 - x1) / (v1 - v2)
```

After calculating `t`, the code checks the following conditions:

- `t % 1 == 0`: This condition ensures that `t` is an integer value. If `t` is not an integer, it means the cats will never meet at an exact point (they may cross each other, but not meet).
- `t > 0`: This condition ensures that `t` is a positive value, as negative time doesn't make sense in this context.

If both conditions (`t % 1 == 0` and `t > 0`) are satisfied, it means the cats will meet at some point in the future, and the code prints `"YES"`. Otherwise, if either condition is not satisfied, it means the cats will never meet, and the code prints `"NO"`.

There will be two arrays of integers. Determine all integers that satisfy the following two conditions:

1. The elements of the first array are all factors of the integer being considered
2. The integer being considered is a factor of all elements of the second array

These numbers are referred to as being between the two arrays. Determine how many such numbers exist.

Example

$a = [2, 6]$

$b = [24, 36]$

There are two numbers between the arrays: 6 and 12.

$6\%2 = 0$, $6\%6 = 0$, $24\%6 = 0$ and $36\%6 = 0$ for the first value.

$12\%2 = 0$, $12\%6 = 0$ and $24\%12 = 0$, $36\%12 = 0$ for the second value. Return 2.

https://www.hackerrank.com/challenges/between-two-sets/problem?isFullScreen=false

Though this problem can be solved with a brute force approach, there's a faster, easier way!

Observations

- All numbers in A evenly divide x if and only if x is divisible by the Least Common Multiple (LCM) of all numbers in A. Let's denote the LCM of A as $factor$.
- x evenly divides all numbers in B if and only if x divides the Greatest Common Divisor (GCD) of all numbers in B. Let's denote the GCD of B as $multiple$.

Approach

Let's find the number of x values satisfying $x \bmod factor = 0$ and $multiple \bmod x = 0$.

- If $multiple$ is not divisible by $factor$, no such x exists.
- If some x exists, we can divide $multiple$, x, and $factor$ by $factor$ (i.e., $multiple \bmod factor = 0$, $x \bmod factor = 0$, and $factor \bmod factor = 0$ are all true). Now we just need to find the number of divisors of $\frac{multiple}{factor}$, which we can do in $O(\sqrt{C})$ time (or faster), where C is the maximum number in sets A and B.

Tips

- Be sure to be careful in calculating $factor$, as this number can be quite large.
- If $factor$ becomes greater than C, then we need stop calculating and say that our answer is zero.

```
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int lcm(int a, int b) {
    return (a / gcd(a, b)) * b;
}

int getTotalX(int a_count, int* a, int b_count, int* b) {
// Find LCM of array a
    int lcm_a = a[0];
    for (int i = 1; i < a_count; i++) {
        lcm_a = lcm(lcm_a, a[i]);
    }

    // Find GCD of array b
    int gcd_b = b[0];
    for (int i = 1; i < b_count; i++) {
        gcd_b = gcd(gcd_b, b[i]);
    }

    // Count numbers between a and b
```

```
    int count = 0;
    for (int i = lcm_a; i <= gcd_b; i += lcm_a) {
        if (gcd_b % i == 0) {
            count++;
        }
    }

    return count;
}
```

Prepare > Mathematics > Geometry > Points On a Line

Points On a Line ★

Problem | Submissions | Leaderboard | Discussions | Editorial

Given n two-dimensional points in space, determine whether they lie on some vertical or horizontal line. If yes, print YES; otherwise, print NO.

Input Format

The first line contains a single positive integer, n, denoting the number of points.
Each line i of n subsequent lines contain two space-separated integers detailing the respective values of x_i and y_i (i.e., the coordinates of the i^{th} point).

Constraints

- $2 \leq n \leq 10$
- $-10 \leq x_i, y_i \leq 10$

Output Format

Print YES if all points lie on some horizontal or vertical line; otherwise, print NO.

https://www.hackerrank.com/challenges/points-on-a-line/problem

The points lie on the same horizontal or vertical line if and only if either of the following conditions holds:

1. $x_u = x_v \forall 1 \leq u, v \leq n$
2. $y_u = y_v \forall 1 \leq u, v \leq n$

We need to print YES if at least one of these conditions is true; otherwise, we print NO.

Code to check if all points line on same line (won't pass testcase since line can be slanting here instead of horizontal or vertical)-

```
for n_itr in range(n):
    xy = input().split()
    x = int(xy[0])
    y = int(xy[1])
    points.append((x,y))

if len(points) <= 2:
    print('YES')  # Two or fewer points always form a line
else:
    flag=0
    x0, y0 = points[0]
    x1, y1 = points[1]
    for i in range(2, len(points)):
        x, y = points[i]
        if (y1 - y0) * (x - x0) != (y - y0) * (x1 - x0):
            print('NO')
            flag=1
            break
    if flag==0:
        print('YES')
```

Correct code-

```
if __name__ == '__main__':
    n = int(input())
    points=[]

    for n_itr in range(n):
        xy = input().split()
        x = int(xy[0])
        y = int(xy[1])
        points.append((x,y))

    if len(points) <= 1:
        print('YES')  # 1 or fewer points always form a line
    else:
        flag=0
        x0, y0 = points[0]
        for i in range(1, len(points)):
            x, y = points[i]
```

```
        if x0!=x and y0!=y:
            print('NO')
            flag=1
            break
    if flag==0:
        print('YES')
```

Consider two points, $p = (p_x, p_y)$ and $q = (q_x, q_y)$. We consider the inversion or point reflection, $r = (r_x, r_y)$, of point p across poi q to be a 180° rotation of point p around q.

Given n sets of points p and q, find r for each pair of points and print two space-separated integers denoting the respective values of r_x and r_y on a new line.

Function Description

Complete the findPoint function in the editor below.

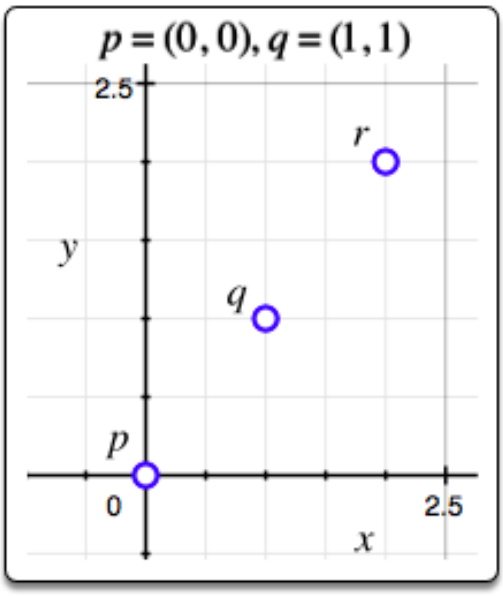
findPoint has the following parameters:

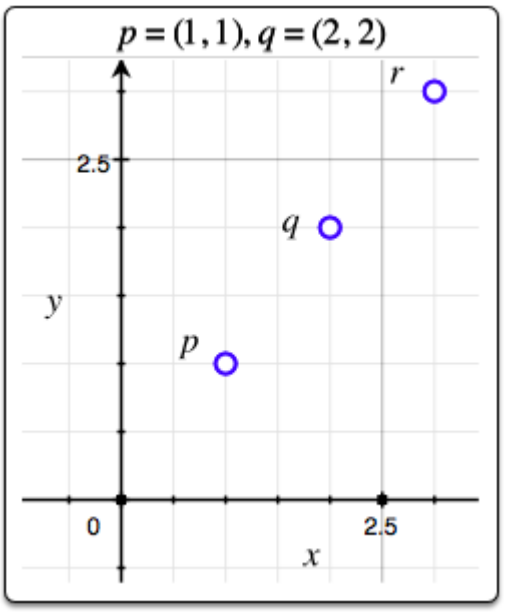
- int px, py, qx, qy: x and y coordinates for points p and q

Returns

- int[2]: x and y coordinates of the reflected point r

https://www.hackerrank.com/challenges/find-point/problem?isFullScreen=false





the formula for the reflection of **a** across the point **p** is

$$\text{Ref}_p(a) = 2p - a.$$

```c
int* findPoint(int px, int py, int qx, int qy, int* result_count) {
    int *arr;
    arr=(int *)malloc(2 * sizeof(int));

    arr[0]=2*qx-px;
    arr[1]=2*qy-py;
    *result_count = 2; //you only have to read existing code and return this

    return arr;
}
```

A person is getting ready to leave and needs a pair of matching socks. If there are n colors of socks in the drawer, how many socks need to be removed to be certain of having a matching pair?

Example $n = 2$

There are 2 colors of socks in the drawer. If they remove 2 socks, they may not match. The minimum number to insure success is 3.

Function Description

Complete the maximumDraws function in the editor below.

maximumDraws has the following parameter:

- int n: the number of colors of socks

Returns

- int: the minimum number of socks to remove to guarantee a matching pair.

https://www.hackerrank.com/challenges/maximum-draws/problem?isFullScreen=false

```c
int maximumDraws(int n) {
   return n+1;
}
```

Two children, Lily and Ron, want to share a chocolate bar. Each of the squares has an integer on it.

Lily decides to share a contiguous segment of the bar selected such that:

- The length of the segment matches Ron's birth month, and,
- The sum of the integers on the squares is equal to his birth day.

Determine how many ways she can divide the chocolate.

Example

$s = [2, 2, 1, 3, 2]$

$d = 4$

$m = 2$

Lily wants to find segments summing to Ron's birth day, $d = 4$ with a length equalling his birth month, $m = 2$. In this case, there are two segments meeting her criteria: $[2, 2]$ and $[1, 3]$.

https://www.hackerrank.com/challenges/the-birthday-bar/problem?isFullScreen=false

Sample Input 0

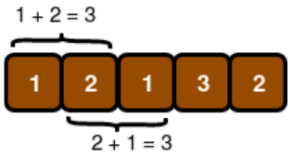
```
5
1 2 1 3 2
3 2
```

Sample Output 0

```
2
```

Explanation 0

Lily wants to give Ron $m = 2$ squares summing to $d = 3$. The following two segments meet the criteria:




```c
int birthday(int s_count, int* s, int d, int m) {
    int cnt=0, window=0;

    //my first window of size m
    for(int i=0; i<m; i++){
        window+=s[i];
    }
    if(window==d) cnt++;

    //to process remaining elements, slide the window
    for(int i=m;i<s_count;i++){
        window-=s[i-m]; //leaving element
        window+=s[i]; //new element

        if(window==d) cnt++;
    }
    return cnt;
}
```

At the annual meeting of Board of Directors of Acme Inc. If everyone attending shakes hands exactly one time with every other attendee, how many handshakes are there?

Example

$n = 3$

There are 3 attendees, $p1$, $p2$ and $p3$. $p1$ shakes hands with $p2$ and $p3$, and $p2$ shakes hands with $p3$. Now they have all shaken hands after 3 handshakes.

Function Description

Complete the handshakes function in the editor below.

handshakes has the following parameter:

- int n: the number of attendees

Returns

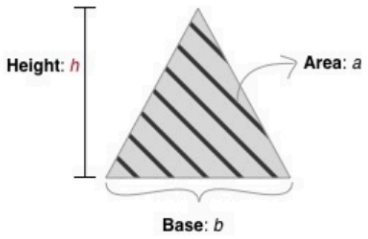
- int: the number of handshakes

https://www.hackerrank.com/challenges/handshake/problem?isFullScreen=false

note that it's (n-1) and not (n+1) used for first n natural numbers sum-

```
int handshake(int n) {
    return n*(n-1)/2;
}
```

Given integers b and a, find the smallest integer h, such that there exists a triangle of height h, base b, having an area of at least a.



Example

$b = 4$

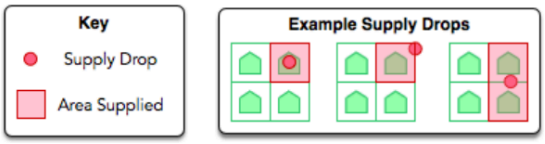
$a = 6$

The minimum height h is 3. One example is a triangle formed at points (0, 0), (4, 0), (2, 3).

https://www.hackerrank.com/challenges/lowest-triangle/problem?isFullScreen=false

```
int lowestTriangle(int trianglebase, int area) {
    //since least height is needed and '/' operator will return like 10
    //instead of 10.12 something we first check divisibility using
    //area=(1/2)*base*height
    if((2*area%trianglebase)==0) return (2*area/trianglebase); //we won't get fraction
    else return (2*area/trianglebase) +1; //return nearest greater whole number

}
```

Luke is daydreaming in Math class. He has a sheet of graph paper with n rows and m columns, and he imagines that there is an army base in each cell for a total of $n \cdot m$ bases. He wants to drop supplies at strategic points on the sheet, marking each drop point with a red dot. If a base contains at least one package inside or on top of its border fence, then it's considered to be supplied. For example:



Given n and m, what's the minimum number of packages that Luke must drop to supply all of his bases?

Example

$n = 2$

$m = 3$

Packages can be dropped at the corner between cells (0, 0), (0, 1), (1, 0) and (1, 1) to supply 4 bases. Another package can be dropped at a border between (0, 2) and (1, 2). This supplies all bases using 2 packages.

https://www.hackerrank.com/challenges/game-with-cells/problem?isFullScreen=false

Sample Input 0

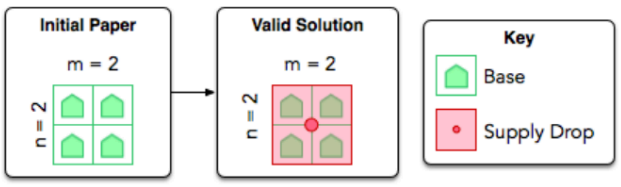
```
2 2
```

Sample Output 0

```
1
```

Explanation 0

Luke has four bases in a 2×2 grid. If he drops a single package where the walls of all four bases intersect, then those four cells can access the package:



Because he managed to supply all four bases with a single supply drop, we print 1 as our answer.

```
def gameWithCells(n, m):
    return math.ceil(n/2) * math.ceil(m/2)
```

Understanding the Problem:

- You have an nxm grid of cells.

- A single package can supply all four cells it touches if placed at a corner where four cells meet.
- If placed on a border between two cells, it can supply both cells.

Optimal Placement:

- To minimize the number of packages, place packages at every other row and column intersection.
- Essentially, each package should be able to cover a 2×2 block of cells.

Mathematical Representation:

- To determine the number of packages needed, you divide the grid into 2×2 blocks.
- For rows (n), you'll need ceil⌈n/2⌉ blocks.
- For columns (m), you'll need ceil⌈m/2⌉ blocks.

We use ceil function because if coverage n/2 is like 2.5 then we need 3 packages

Leonardo loves primes and created q queries where each query takes the form of an integer, n. For each n, count the maximum number of distinct prime factors of any number in the inclusive range $[1, n]$.

Note: Recall that a prime number is only divisible by 1 and itself, and 1 is not a prime number.

Example

$n = 100$

The maximum number of distinct prime factors for values less than or equal to 100 is 3. One value with 3 distinct prime factors is 30. Another is 42.

Function Description

Complete the primeCount function in the editor below.

primeCount has the following parameters:

- int n: the inclusive limit of the range to check

Returns

- int: the maximum number of distinct prime factors of any number in the inclusive range $[0 - n]$.

https://www.hackerrank.com/challenges/leonardo-and-prime/problem?isFullScreen=false

```
6
1
2
3
500
5000
10000000000
```

Sample Output

```
0
1
1
4
5
10
```

Explanation

1. 1 is not prime and its only factor is itself.
2. 2 has 1 prime factor, 2.
3. The number 3 has 1 prime factor, 3, 2 has 1 and 1 has 0 prime factors.
4. The product of the first four primes is $2 \times 3 \times 5 \times 7 = 210$. While higher value primes may be a factor of some numbers, there will never be more than 4 distinct prime factors for a number in this range.

To find the maximum count of distinct prime factors for any number in the inclusive range $[1, n]$, take the products of prime numbers starting from 2 and continue until the product becomes greater than the value of n. The total number of prime numbers included in this product is the answer. The fastest way to do this is to create an array of primes from 0 through 47. The product of these numbers is $614,889,782,588,491,410$ which is greater than the constraint for n. While it is possible that a number in the range has a higher prime factor than 47, it will never have more of them.

```
prime[] = {2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47};

product = 1
answer = 0

for i = 0 to length(prime) - 1
      product = product * prime[i];
      if(product <= n)
            answer++;
print answer
```

```
q = input()

primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]

i = 0
while i < int(q):
    n = int(input())
    maxFacts = -1
    product = 1
    k = 0
    while n >= product:
        maxFacts += 1
        product = product * primes[k]
        k+=1
    print (maxFacts)
    i+=1
```

Marie invented a Time Machine and wants to test it by time-traveling to visit Russia on the Day of the Programmer (the 256th day of the year) during a year in the inclusive range from 1700 to 2700.

From 1700 to 1917, Russia's official calendar was the Julian calendar; since 1919 they used the Gregorian calendar system. The transition from the Julian to Gregorian calendar system occurred in 1918, when the next day after January 31st was February 14th. This means that in 1918, February 14th was the 32nd day of the year in Russia.

In both calendar systems, February is the only month with a variable amount of days; it has 29 days during a leap year, and 28 days during all other years. In the Julian calendar, leap years are divisible by 4; in the Gregorian calendar, leap years are either of the following:

- Divisible by 400.
- Divisible by 4 and not divisible by 100.

Given a year, y, find the date of the 256th day of that year according to the official Russian calendar during that year. Then print it in the format `dd.mm.yyyy`, where `dd` is the two-digit day, `mm` is the two-digit month, and `yyyy` is y.

For example, the given $year$ = 1984. 1984 is divisible by 4, so it is a leap year. The 256th day of a leap year after 1918 is September 12, so the answer is `12.09.1984`.

Function Description

Complete the dayOfProgrammer function in the editor below. It should return a string representing the date of the 256th day of the year given.

dayOfProgrammer has the following parameter(s):

- year: an integer

https://www.hackerrank.com/challenges/day-of-the-programmer/problem?isFullScreen=false

```python
def dayOfProgrammer(year):
    if year == 1918:
        return "26.09.1918"

    def isLeapYear(y):
        if y <= 1917:
            return y % 4 == 0
        else:
            return y % 400 == 0 or (y % 4 == 0 and y % 100 != 0)

    if isLeapYear(year):
        day = 256 - 244
    else:
        day = 256 - 243

    month = 9  # September

    return f"{day:02d}.{month:02d}.{year}"
```

First, we need to determine which calendar system to use based on the year:

- Julian calendar for years 1700 to 1917
- Special case for 1918
- Gregorian calendar for years 1919 to 2700

Then, we need to determine if it's a leap year:

- For Julian calendar: year divisible by 4
- For Gregorian calendar: year divisible by 400, or (divisible by 4 and not by 100)

Calculate the 256th day of the year based on whether it's a leap year or not.

Convert this to a date in the format dd.mm.yyyy.

First, why September (month = 9)?

- September is the 9th month of the year.
- The 256th day of the year always falls in September, regardless of whether it's a leap year or not.

Now, let's understand the 244 and 243:

- In a non-leap year: January (31) + February (28) + March (31) + April (30) + May (31) + June (30) + July (31) + August (31) = 243 days
- In a leap year: January (31) + February (29) + March (31) + April (30) + May (31) + June (30) + July (31) + August (31) = 244 days

So, the calculation 256 - 244 or 256 - 243 is determining which day in September is the 256th day of the year:

- In a leap year: 256 - 244 = 12, so it's September 12th
- In a non-leap year: 256 - 243 = 13, so it's September 13th

The `:02d` in the f-string is a formatting specifier. Let's break it down:

- `:` starts the format specifier
- `0` is a fill character (padding with zeros)
- `2` specifies the width (number of characters)
- `d` indicates we're formatting an integer (decimal)

So, `{day:02d}` means:

- Format `day` as an integer
- Ensure it takes up at least 2 characters
- If it's less than 2 digits, pad with 0 on the left

This ensures that single-digit days (1-9) are displayed as '01', '02', '03', etc., instead of just '1', '2', '3'.

The same applies to `{month:02d}`, ensuring months are always two digits (01 for January, 02 for February, etc.).

For example:

- If `day = 5` and `month = 9`, it will format as "05.09.yyyy"
- If `day = 12` and `month = 10`, it will format as "12.10.yyyy"

Cities on a map are connected by a number of roads. The number of roads between each city is in an array and city 0 is the starting location. The number of roads from city 0 to city 1 is the first value in the array, from city 1 to city 2 is the second, and so on.

How many paths are there from city 0 to the last city in the list, modulo 1234567?

Example

$n = 4$

$routes = [3, 4, 5]$

There are 3 roads to city 1, 4 roads to city 2 and 5 roads to city 3. The total number of roads is $3 \times 4 \times 5 \mod 1234567 = 60$.

Note

Pass all the towns T_i for i=1 to n-1 in numerical order to reach T_n.

Function Description

Complete the connectingTowns function in the editor below.

connectingTowns has the following parameters:

- int n: the number of towns
- int routes[n-1]: the number of routes between towns

Returns

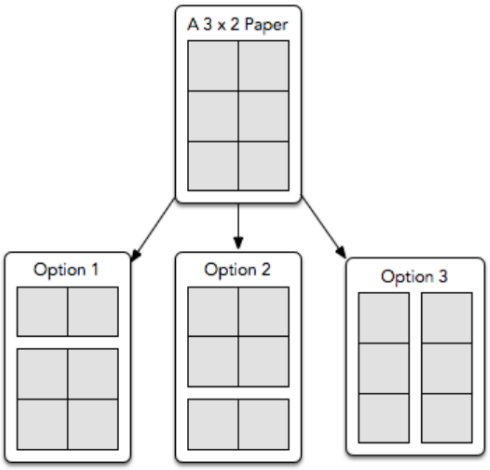
- int: the total number of routes, modulo 1234567.

https://www.hackerrank.com/challenges/connecting-towns/problem?isFullScreen=false

```
def connectingTowns(n, routes):
    # Write your code here
    mod=1234567
    prod=1

    for i in routes:
        prod*=i
    return prod%mod
```

Mary has an $n \times m$ piece of paper that she wants to cut into 1×1 pieces according to the following rules:

- She can only cut one piece of paper at a time, meaning she cannot fold the paper or layer already-cut pieces on top of one another.
- Each cut is a straight line from one side of the paper to the other side of the paper. For example, the diagram below depicts the three possible ways to cut a 3×2 piece of paper:



Given n and m, find and print the minimum number of cuts Mary must make to cut the paper into $n \cdot m$ squares that are 1×1 unit in size.

https://www.hackerrank.com/challenges/p1-paper-cutting/problem?isFullScreen=false

```
def solve(n, m):
    # Write your code here
    return m*n-1
```

it looks like m+n-2 but actually it's m*n-1

There is a sequence whose n^{th} term is

$$T_n = n^2 - (n-1)^2$$

Evaluate the series

$$S_n = T_1 + T_2 + T_3 + \cdots + T_n$$

Find $S_n \bmod (10^9 + 7)$.

Example

$n = 3$

The series is $1^2 - 0^2 + 2^2 - 1^2 + 3^2 - 2^2 = 1 + 3 + 5 = 9$.

Function Description

Complete the summingSeries function in the editor below.

summingSeries has the following parameter(s):

- int n: the inclusive limit of the range to sum

Returns

- int: the sum of the sequence, modulo $(10^9 + 7)$

https://www.hackerrank.com/challenges/summing-the-n-series/problem

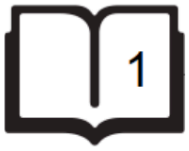
```
def summingSeries(n):
    mod = 10**9 + 7
    return (n * n) % mod
```

Current approach: Your current implementation is using a loop to calculate the sum, which has a time complexity of O(n). For large values of n, this can indeed cause a TLE error.

Mathematical approach: There's a mathematical pattern we can exploit here. Let's look at the series: (1^2 - 0^2) + (2^2 - 1^2) + (3^2 - 2^2) + ... + (n^2 - (n-1)^2) This telescopes. Each term cancels out except for the last n^2: = n^2

Optimized solution: We can simply return n^2 % mod. This reduces the time complexity to O(1).

A teacher asks the class to open their books to a page number. A student can either start turning pages from the front of the book or from the back of the book. They always turn pages one at a time. When they open the book, page 1 is always on the right side:



When they flip page 1, they see pages 2 and 3. Each page except the last page will always be printed on both sides. The last page may only be printed on the front, given the length of the book. If the book is n pages long, and a student wants to turn to page p, what is the minimum number of pages to turn? They can start at the beginning or the end of the book.

Given n and p, find and print the minimum number of pages that must be turned in order to arrive at page p.

https://www.hackerrank.com/challenges/drawing-book/problem?isFullScreen=false

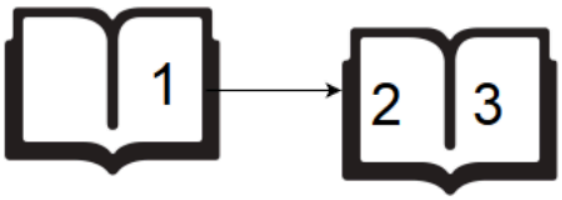
```
6
2
```

Sample Output 0

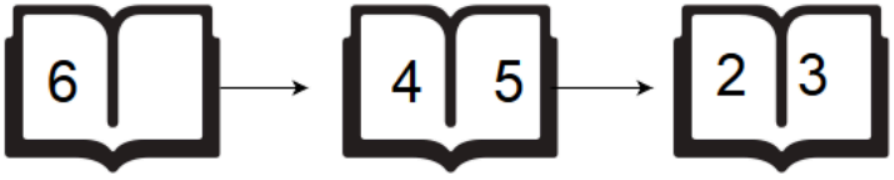
```
1
```

Explanation 0

If the student starts turning from page 1, they only need to turn 1 page:



If a student starts turning from page 6, they need to turn 2 pages:



```python
def pageCount(n, p):
    # Turns from the front
    front_turns = p // 2

    # Turns from the back
    back_turns = n // 2 - p // 2

    # Return the minimum of the two
    return min(front_turns, back_turns)
```

Let's consider the turns from the front of the book:

- Page 1 is already visible, so we start turning from page 2.
- Every turn reveals 2 new pages.
- To reach page p, we need to turn (p / 2) pages, rounded down.

Now, let's consider the turns from the back of the book:

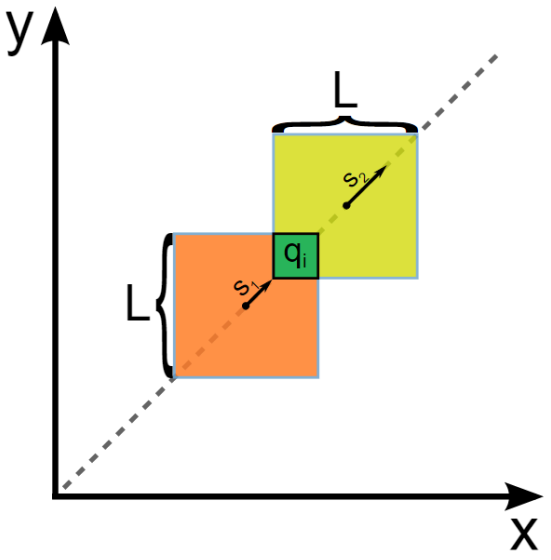
- If the book has n pages, we start from page n or n-1 (depending on whether n is odd or even).
- We need to turn until we reach or pass page p.
- The number of turns from the back is (n / 2) - (p / 2), rounded down.

The minimum number of turns will be the smaller of these two values.

Sherlock is given 2 square tiles, initially both of whose sides have length l placed in an $x - y$ plane. Initially, the bottom left corners of each square are at the origin and their sides are parallel to the axes.

At $t = 0$, both squares start moving along line $y = x$ (along the positive x and y) with velocities $s1$ and $s2$.

For each querydetermine the time at which the overlapping area of tiles is equal to the query value, $queries[i]$.



https://www.hackerrank.com/challenges/sherlock-and-moving-tiles/problem?isFullScreen=false

Input Format

First line contains integers $l, s1, s2$.

The next line contains q, the number of queries.

Each of the next q lines consists of one integer $queries[i]$ in one line.

Constraints

$1 \leq l, s1, s2 \leq 10^9$

$1 \leq q \leq 10^5$

$1 \leq queries[i] \leq L^2$

$s1 \neq s2$

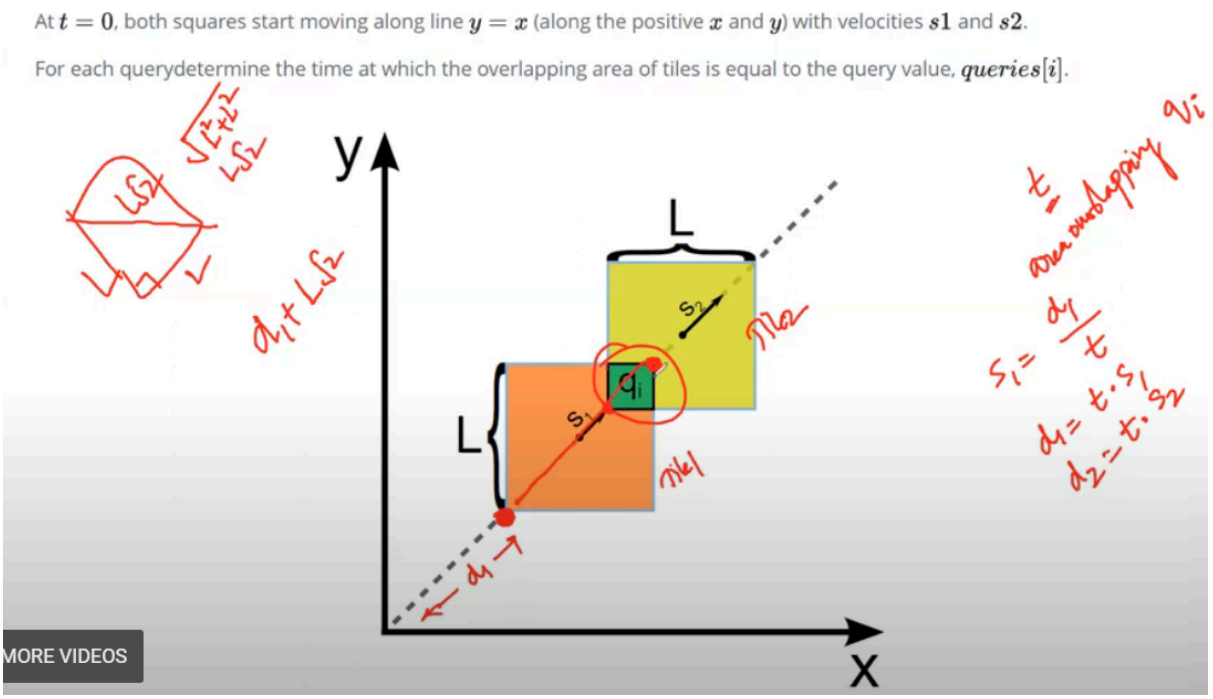
Sample Input

```
10 1 2
2
50
100
```

Sample Output

```
4.1421
0.0000
```

```python
def movingTiles(l, s1, s2, queries):
    no_queries = len(queries)
    ans = []
    for i in range(no_queries):
        ans.append((l * math.sqrt(2) - math.sqrt(queries[i] * 2)) / abs(s1 - s2))
    return ans
```



diagonal of a square is l*(root 2). Area is qi^2 so diagonal length is root of (qi^2 *2)= *qi**(root 2)

An avid hiker keeps meticulous records of their hikes. During the last hike that took exactly $steps$ steps, for every step it was noted if it was an uphill, U, or a downhill, D step. Hikes always start and end at sea level, and each step up or down represents a 1 unit change in altitude. We define the following terms:

- A mountain is a sequence of consecutive steps above sea level, starting with a step up from sea level and ending with a step down to sea level.
- A valley is a sequence of consecutive steps below sea level, starting with a step down from sea level and ending with a step up to sea level.

Given the sequence of up and down steps during a hike, find and print the number of valleys walked through.

Example

$steps = 8$ $path = [DDUUUUDD]$

The hiker first enters a valley 2 units deep. Then they climb out and up onto a mountain 2 units high. Finally, the hiker returns to sea level and ends the hike.

Function Description

Complete the countingValleys function in the editor below.

countingValleys has the following parameter(s):

- int steps: the number of steps on the hike
- string path: a string describing the path

https://www.hackerrank.com/challenges/counting-valleys/problem?isFullScreen=false

Sample Input

```
8
UDDDUDUU
```

Sample Output

```
1
```

Explanation

If we represent `_` as sea level, a step up as `/`, and a step down as `\`, the hike can be drawn as:

```
_/\      _
   \    /
    \/\/
```

The hiker enters and leaves one valley.

```
def countingValleys(steps, path):
    cnt=0
    level=0 #initially sea level
    valley=0 #means we're in valley

    for i in path:
        if i=='U':
            level+=1
        else:
            level-=1

        if level<0 and valley==0: #went below sea level so valley
            cnt+=1
            valley=1

        elif level==0 and valley==1:
            valley=0 #came back to sea level so valley crossed

    return cnt
```

Given a set of distinct integers, print the size of a maximal subset of S where the sum of any 2 numbers in S' is not evenly divisible by k

.

Example

$S = [19, 10, 12, 10, 24, 25, 22]$ $k = 4$

One of the arrays that can be created is $S'[0] = [10, 12, 25]$. Another is $S'[1] = [19, 22, 24]$. After testing all permutations, the maximum length solution array has 3 elements.

Function Description

Complete the nonDivisibleSubset function in the editor below.

nonDivisibleSubset has the following parameter(s):

- int S[n]: an array of integers
- int k: the divisor

Returns

- int: the length of the longest subset of S meeting the criteria

Input Format

The first line contains 2 space-separated integers, n and k, the number of values in S and the non factor.

The second line contains n space-separated integers, each an $S[i]$, the unique values of the set.

https://www.hackerrank.com/challenges/non-divisible-subset/problem?isFullScreen=false

Here,we can form a maximal subset of S as **S`= [ 3 , 1, 4 ].** To verify we can ***take all possible pairs of elements*** from S` and check if the sum of any two elements is evenly divisible by k=3 or not.
Here 3 + 1 = 4 which is not evenly divisible by 3. Also 3 + 4 = 7 is also not evenly divisible by 3. And another possible pair we can check is 1+4 =5 which too is not evenly divisible by 3. So it satisfies the condition !
In this example, there is no way to form a subset with more than 3 elements which satisfies the condition.
Here,what ***we are expected to find is the size of S`*** and not what it contains. So in this example, our ***answer will be 3*** as S' contains three elements.
Note that we could have made another subset as S`= [3, 2, 5] which also satisfies the condition and it too has 3 elements. So **we can have multiple subsets of a certain maximum size** which satisfy the condition but we are concerned only with the size and not the subsets themselves.

https://medium.com/@mrunankmistry52/non-divisible-subset-problem-comprehensive-explanation-c878a752f057

```
def nonDivisibleSubset(k, s):
    count = [0] * k

    for i in s:
        remainder = i % k
        count[remainder] +=1

    ans = min( count[0] , 1)        # Handling case 1

    if k % 2 == 0:                  # Handling case even exception case
        ans += min(count[k//2] ,1 )

    for i in range( 1 , k//2 + 1):    # Check for the pairs and take appropriate count
        if i != k - i:          # Avoid over-counting when k is even
            ans += max(count[i] , count[k-i])
    return ans
```

Generate Pythagorean Triplets

Last Updated : 12 Oct, 2023

A Pythagorean triplet is a set of three positive integers a, b and c such that $a^2 + b^2 = c^2$. Given a limit, generate all Pythagorean Triples with values smaller than given limit.

```
Input : limit = 20
Output : 3 4 5
         8 6 10
         5 12 13
         15 8 17
         12 16 20
```

An **Efficient Solution** can print all triplets in O(k) time where k is number of triplets printed. The idea is to use square sum relation of Pythagorean triplet, i.e., addition of squares of a and b is equal to square of c, we can write these number in terms of m and n such that,

$$a = m^2 - n^2$$
$$b = 2 * m * n$$
$$c = m^2 + n^2$$

because,

$$a^2 = m^4 + n^4 - 2 * m^2 * n^2$$
$$b^2 = 4 * m^2 * n^2$$
$$c^2 = m^4 + n^4 + 2* m^2 * n^2$$

```
# Function to generate pythagorean
# triplets smaller than limit
def pythagoreanTriplets(limits) :
    c, m = 0, 2

    # Limiting c would limit
    # all a, b and c
    while c < limits :

        # Now loop on n from 1 to m-1
        for n in range(1, m) :
            a = m * m - n * n
            b = 2 * m * n
            c = m * m + n * n

            # if c is greater than
            # limit then break it
            if c > limits :
                break

            print(a, b, c)

        m = m + 1
```

Kristen loves playing with and comparing numbers. She thinks that if she takes two different positive numbers, the one whose digits sum to a larger number is better than the other. If the sum of digits is equal for both numbers, then she thinks the smaller number is better. For example, Kristen thinks that 13 is better than 31 and that 12 is better than 11.

Given an integer, n, can you find the divisor of n that Kristin will consider to be the best?

https://www.hackerrank.com/challenges/best-divisor/problem?isFullScreen=false

Sample Input 0

```
12
```

Sample Output 0

```
6
```

Explanation 0

The set of divisors of 12 can be expressed as $\{1, 2, 3, 4, 6, 12\}$. The divisor whose digits sum to the largest number is 6 (which, having only one digit, sums to itself). Thus, we print 6 as our answer.

```python
def sum_of_digits(num):
    return sum(int(digit) for digit in str(num))

if __name__ == '__main__':
    n = int(input().strip())

    best = 1
    best_sum = 1

    for i in range(1, n + 1):
        if n % i == 0:
            current_sum = sum_of_digits(i)

            if current_sum > best_sum or (current_sum == best_sum and i < best):
                best = i
                best_sum = current_sum

    print(best)
```

A person wants to determine the most expensive computer keyboard and USB drive that can be purchased with a give budget. Given price lists for keyboards and USB drives and a budget, find the cost to buy them. If it is not possible to buy both items, return -1.

Example

$b = 60$

$keyboards = [40, 50, 60]$

$drives = [5, 8, 12]$

The person can buy a 40 **keyboard** $+ 12$ **USB drive** $= 52$, or a 50 **keyboard** $+ 8$ **USB drive** $= 58$. Choose the latter as the mor expensive option and return 58.

Function Description

Complete the getMoneySpent function in the editor below.

getMoneySpent has the following parameter(s):

- int keyboards[n]: the keyboard prices
- int drives[m]: the drive prices
- int b: the budget

Returns

- int: the maximum that can be spent, or -1 if it is not possible to buy both items

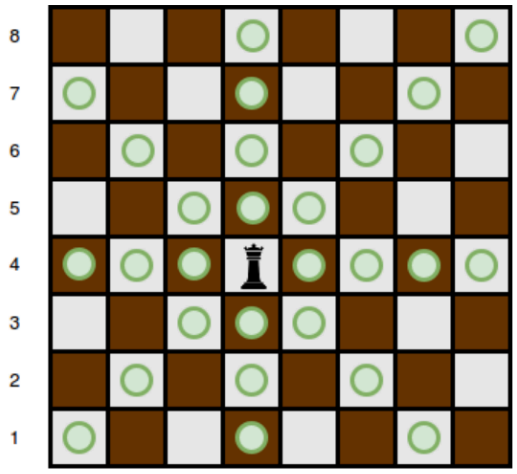
https://www.hackerrank.com/challenges/electronics-shop/problem?isFullScreen=false

```
def getMoneySpent(keyboards, drives, b):
    max_cost = -1

    for keyboard in keyboards:
        for drive in drives:
            total_cost = keyboard + drive
            if total_cost <= b and total_cost > max_cost:
                max_cost = total_cost

    return max_cost
```

You will be given a square chess board with one queen and a number of obstacles placed on it. Determine how many squares the queen can attack.

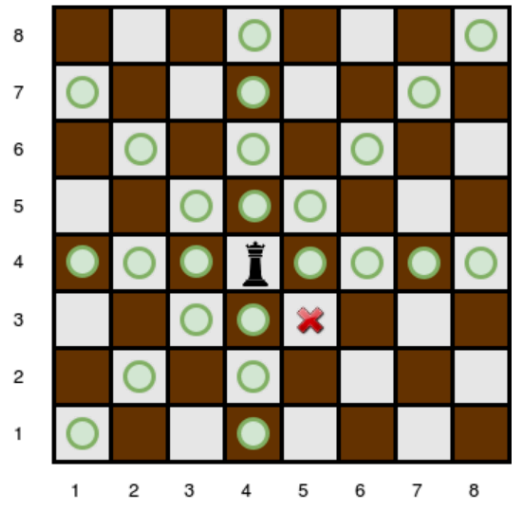
A queen is standing on an $n \times n$ chessboard. The chess board's rows are numbered from 1 to n, going from bottom to top. Its columns are numbered from 1 to n, going from left to right. Each square is referenced by a tuple, (r, c), describing the row, r, and column, c, where the square is located.

The queen is standing at position (r_q, c_q). In a single move, she can attack any square in any of the eight directions (left, right, up, down, and the four diagonals). In the diagram below, the green circles denote all the cells the queen can attack from $(4, 4)$:



https://www.hackerrank.com/challenges/queens-attack-2/problem?isFullScreen=false

There are obstacles on the chessboard, each preventing the queen from attacking any square beyond it on that path. For example, an obstacle at location $(3, 5)$ in the diagram above prevents the queen from attacking cells $(3, 5)$, $(2, 6)$, and $(1, 7)$:



Given the queen's position and the locations of all the obstacles, find and print the number of squares the queen can attack from her position at (r_q, c_q). In the board above, there are 24 such squares.

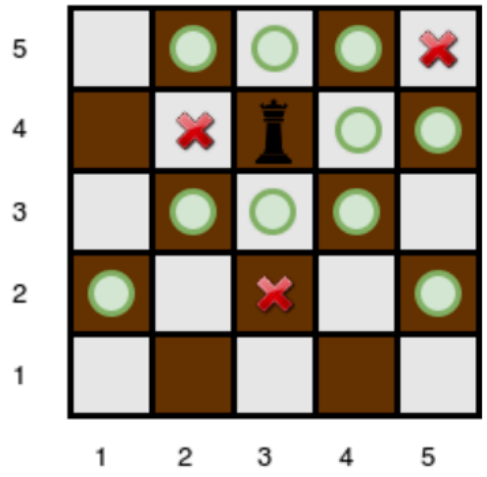
```
5 3
4 3
5 5
4 2
2 3
```

Sample Output 1

```
10
```

Explanation 1

The queen is standing at position $(4, 3)$ on a 5×5 chessboard with $k = 3$ obstacles:



```python
def isValid(x,y,n):
    if x>0 and x<=n and y>0 and y<=n:
        return True
    return False

def queensAttack(n, k, r_q, c_q, obstacles):
    # Write your code here
    dir=[[1,-1],[1,0],[1,1],[0,1],[-1,1],[-1,0],[-1,-1],[0,-1]]
    cnt=0

    for d in dir:
        x,y=r_q, c_q
        while True:
            x+=d[0]
            y+=d[1]

            if isValid(x,y,n) and [x,y] not in obstacles:
                cnt+=1
            else:
                break
    return cnt
```

above code gives TLE error for large testcases

$O(K)$ Approach:

The key idea for this approach is that you can iterate over the obstacles and for those that are in the queen's path, calculate the free cells up to that obstacle. If there is no obstacle in the path you have to calculate the number of free cells up to the edge of the board in that direction.

For any (x_1, y_1) and x_2, y_2:

- If they are in the same row: abs($x_1 - x_2 - 1$) is the number of free cells between them.
- If they are in the same column: abs($y_1 - y_2 - 1$) is the number of free cells between them.
- If they are diagonal, either abs($x_1 - x_2 - 1$) or abs($y_1 - y_2 - 1$) is the number of free cells between them.

def queensAttack(N, K, x, y, obstacles):

diagonals (x,y is queen's coordinates)

d11 = min(x - 1, y - 1)

d12 = min(N - x, N - y)

d21 = min(N - x, y - 1)

d22 = min(x - 1, N - y)

```
# rows and cols
r1 = y - 1
r2 = N - y
c1 = x - 1
c2 = N - x

for i in range(K):
    ox,oy=obstacles[i]
    if x > ox and y > oy and x - ox == y - oy:
        d11 = min(d11, x - ox - 1)
    if ox > x and oy > y and ox - x == oy - y:
        d12 = min(d12, ox - x - 1)
    if ox > x and y > oy and ox - x == y - oy:
        d21 = min(d21, ox - x - 1)
    if x > ox and oy > y and x - ox == oy - y:
        d22 = min(d22, x - ox - 1)

    if x == ox and oy < y:
        r1 = min(r1, y - oy - 1)
    if x == ox and oy > y:
        r2 = min(r2, oy - y - 1)
    if y == oy and ox < x:
        c1 = min(c1, x - ox - 1)
    if y == oy and ox > x:
        c2 = min(c2, ox - x - 1)

return d11 + d12 + d21 + d22 + r1 + r2 + c1 + c2
```

`d11 = min(x - 1, y - 1)`: This represents the maximum number of moves diagonally up-left. It's limited by either the distance to the left edge (x - 1) or the top edge (y - 1), whichever is smaller.

`d12 = min(N - x, N - y)`: This is for diagonal moves down-right. It's limited by the distance to the right edge (N - x) or bottom edge (N - y).

`d21 = min(N - x, y - 1)`: For diagonal moves up-right. Limited by distance to right edge (N - x) or top edge (y - 1).

`d22 = min(x - 1, N - y)`: For diagonal moves down-left. Limited by distance to left edge (x - 1) or bottom edge (N - y).

`r1 = y - 1`: Maximum moves straight up.

`r2 = N - y`: Maximum moves straight down.

`c1 = x - 1`: Maximum moves straight left.

`c2 = N - x`: Maximum moves straight right.

Martha is interviewing at Subway. One of the rounds of the interview requires her to cut a bread of size $l \times b$ into smaller identical pieces such that each piece is a square having maximum possible side length with no left over piece of bread.

Input Format

The first line contains an integer T. T lines follow. Each line contains two space separated integers l and b which denote length and breadth of the bread.

https://www.hackerrank.com/challenges/restaurant/problem?isFullScreen=false

Sample Input 0

```
2
2 2
6 9
```

Sample Output 0

```
1
6
```

Explanation 0

The 1st testcase has a bread whose original dimensions are 2×2, the bread is uncut and is a square. Hence the answer is 1.

The 2nd testcase has a bread of size 6×9. We can cut it into 54 squares of size 1×1, 6 of size 3×3. For other sizes we will have leftovers. Hence, the number of squares of maximum size that can be cut is 6.

The question asks on how large can a square piece be to fit into the original area **without any** remainder. When we take another example with a bread of size 10×15 it is clear that to be square the largest common factor we can find is 5 such that we can order 2×3 squares of size 5. This is everything needed to solve the problem:

$$\begin{aligned} R(l,b) &= \frac{l}{\gcd(l,b)} \cdot \frac{b}{\gcd(l,b)} \\ &= \frac{l \cdot b}{\gcd(l,b)^2} \end{aligned}$$

```python
def restaurant(l, b):
    return (l * b) // (math.gcd(l, b) ** 2)
```