TCS Code Vita Season-9 SET-3 (8th August)

Count Pairs

Problem Description

Given an array of integers A, and an integer K find number of happy elements.

Element X is happy if there exists at least 1 element whose difference is less than K i.e. an element X is happy, if there is another element in the range [X-K, X+K] other than X itself.

Constraints

1 <= N <= 10^5

0 <= K <= 10^5

0 <= A[i] <= 10^9

Input

First line contains two integers N and K where N is size of the array and K is a number as described above

Second line contains N integers separated by space.

Output

Print a single integer denoting the total number of happy elements.

Time

Limit 1

Exampl

es

Exampl

e 1

Input

6 3

5 5 7 9 15 2

Output

5

Explanation

Other than number 15, everyone has at least 1 element in the range [X-3, X+3]. Hence they are all happy elements. Since these five are in number, the output is 5.

Example 2

Input

3 2

1 3 5

Output

3

Explanation

All numbers have at least 1 element in the range [X-2, X+2]. Hence they are all happy elements. Since these three are in number, the output is 3.

##

###

Prime Time Again

Problem

Description

Here on earth, our 24-hour day is composed of two parts, each of 12 hours. Each hour in each part has a corresponding hour in the other part separated by 12 hours: the hour essentially measures the duration since the start of the day part. For example, 1 hour in the first part of the day is equivalent to 13, which is 1 hour into the second part of the day.

Now, consider the equivalent hours that are both prime numbers. We have 3 such instances for a 24- hour 2-part day:

5~17

7~19

11~23

Accept two natural numbers D, P >1 corresponding respectively to number of hours

per day and number of parts in a day separated by a space. D should be divisible by P, meaning that the number of hours per part (D/P) should be a natural number. Calculate the number of instances of equivalent prime hours.
Output zero if there is no such instance. Note that we require each equivalent hour in each part in a day to be a prime number.

Example:

Input: 24 2

Output: 3 (We have 3 instances of equivalent prime hours: 5~17, 7~19 and 11~23.)

Constraint

s 10 <= D

< 500

2 <= P < 50

Input

Single line consists of two space separated integers, D and P corresponding to number of hours per day and number of parts in a day respectively

Output

Output must be a single number, corresponding to the number of instances of equivalent prime number, as described above

Time

Limit 1

Exampl

es

Exampl

e 1

Input

```
36 3
```

Output

```
2
```

Explanation

In the given test case D = 36 and P = 3

Duration of each day part = 12

2~14~X

3~15~X

5~17~29 - instance of equivalent prime hours

7~19~31 - instance of equivalent prime hours

11~23~X

Hence the answers is 2.

Example 2

Input

49 7

Output

```
0
```

Explanation

Duration of each day part = 7

2~9~X~23~X~37~X

3~X~17~X~31~X~X

5~X~19~X~X~X~47

7~X~X~X~X~X~X

Hence there are no equivalent prime hours.

###

Number

Distancing

Problem

Description

Consider 9 natural numbers arranged in a 3x3 matrix:

n11 n12 n13

n21 n22 n23

n31 n32 n33

Define numbers "in contact" with a given number to be those that appear closest to it on the same row, column or diagonally across:

Contacts of number n11: n12, n22 and n21

Contacts of number n12: n11, n21, n22, n23, n13

Contacts of number n13: n12, n22, n23

Contacts of number n21: n11, n12, n22, n32, n31

Contacts of number n22: n11, n12, n13, n23, n33, n32, n31, n21

Contacts of number n23: n13, n12, n22, n32, n33

Contacts of number n31: n21, n22, n32

Contacts of number n32: n31, n21, n22, n23, n33

Contacts of number n33: n32, n22, n23

The problem now is that numbers having a common factor (other than 1) should not be "in contact". In other words, a pair of numbers can remain neighbours only if their highest common factor is 1.

The following rules apply to enforce this "distancing":

1. The central number (n22) stays put.

2. The corner numbers (n11, n13, n33, n31) can move in the same row or column or diagonally away from the centre.

3. The numbers "on the walls" (n12, n23, n32, n21) can only move from the walls i.e. n21 can only move "left", n12 can only move "up", n23 can only move "right" and n32 can only move "down".

4. Each number should stay put as far as possible and the "distancing" operation should result in the least number of numbers ending up without any contacts.

5. After satisfying rule 4, if there are multiple options for the final matrix, then the "distancing" operation should result in the smallest (m x n matrix, including the intervening blank space elements, with the least possible value of m*n).

6. If, after satisfying all the rules above, there are multiple distancing options for a set of numbers, the largest number keeps to its original cell.

Constraints

1 <= Element of grid <= 100

Input

First line consists of 9 space separated integers denoting n11, n12, n13, n23, n33 respectively.

Output

Print the "contact" less numbers in ascending order of their value separated by space. Output "None" if there are no such numbers.

Time

Limit 1

Exampl

es

Exampl

e 1

Input

23 33 12 1 2 5 25 6 10

Output

```
10
```

Explanation

Initial configuration

```
23 33 12

1 2 5

25 6 10
```

The optimal distancing options result in the following possibility (space denoted by *):

```
23 33 * 12

1 2 5 *

25 * * *

* 6 * 10
```

10 ends up as the number without contacts.

Example 2

Input

1 2 3 4 5 6 7 8 9

Output

None

Explanation

Initial configuration:

1 2 3

4 5 6

7 8 9

The optimal distancing options result in the following 5x3 matrix (space denoted by *):

* 2 3

1 * *

4 5 6

7 * *

* 8 9

There is finally no number without a contact.

Example 3

Input

2 6 2 10 19 12 2 20 2

Output

2 2 2 2 10 12

Explanation

Initial Configuration:

Approach 1) Moving 6 and 20

Final Matrix

2 * 6 * 2

* * * * *

* 10 19 12 *

* * * * *

2 * 20 * 2

Approach 2) Moving 10 and 12

2 * * * 2

* * 6 * *

10 * 19 * 12

* * 20 * *

2 * * * 2

We prefer approach 2) and not 1) since the largest of all the elements (20) needs to be retained in it's original cell.

##

Railway Station

Problem

Description

Given schedule of trains and their stoppage time at a Railway Station, find minimum number of platforms needed.

Note -

If Train A's departure time is x and Train B's arrival time is x, then we can't accommodate Train B on the same platform as Train A.

Constraints

1 <= N <= 10^5

0 <= a <= 86400

0 < b <= 86400

Number of platforms > 0

Input

First line contains N denoting number of trains.

Next N line contain 2 integers, a and b, denoting the arrival time and stoppage time of train.

Output

Single integer denoting the minimum numbers of platforms needed to accommodate every train.

Time

Limit 1

Exampl

es

Exampl

e 1

Input

3

10 2

5 10

13 5

Output

2

Explanation

The earliest arriving train at time t = 5 will arrive at platform# 1. Since it will stay there till t = 15, train arriving at time t = 10 will arrive at platform# 2. Since it will depart at time t = 12, train arriving at time t
= 13 will arrive at platform# 2.

Example 2

Input

2

2 4

6 2

Output

2

Explanation

Platform #1 can accommodate train 1.

Platform #2 can accommodate train 2.

Note that the departure of train 1 is same as arrival of train 2, i.e. 6, and thus we need a separate platform to accommodate train 2.

##
####

Lift

Problem Description

In a building there are some lifts. Optimize the allocation of lifts.

Say there are N requests and M lifts. Minimize the maximum request waiting time.

Rules of lift allocation

1) One needs to assign indexes to lifts. i.e. decide lift #1, lift #2, lift #3, etc apriori.

2) Since all N requests are known apriori decide the initial (at time t = 0) location of lift such that it will help in minimizing waiting time.

3) Even if all the N requests time are known apriori, other than initial time, i.e. t = 0, the waiting lifts cannot be moved towards target floor until the request is actually issued.

4) After a request is issued for optimality calculation, even a lift is in transit it can be considered for serving that request.

5) If at any moment, more than one lift can serve the request with same waiting time, give preference to the lift with lower index. i.e. If Lift #2 and Lift #4 can serve a particular request in 2 seconds, then prefer Lift #2 over Lift #4.

6) Neglect the time required to get in and get out of a lift.

7) Once a lift starts serving a request it would not stop in between. If would finally stop at destination of that request.

8) The speed of lift is 1 floor/second.

Constraints

0 <= T <= 86400.

0 <= S, D <= 100.

1 <= N, M <= 1000.

Input

First line contains 2 integers, N and M, representing the number of requests and number of lifts in the building respectively.

Next N lines contain 3 integers, T, S, and D, representing the timestamp of request, source floor, destination floor respectively.

Output

Print single integer representing the waiting time

Time

Limit 1

Exampl

es

Exampl

e 1

Input

3 2

0 2 3

4 2 5

6 7 3

Output

3

Explanation

There are 3 requests and 2 lifts.

Initially at time t = 0, both the lifts, lift #1 and lift #2 will be at floor 2 (Initial allocation).

Request #1 and Request #2 can be responded instantly, i.e. waiting time = 0.

Lift #1 will get unallocated at floor 3 at time t = 1.

Lift #1 can move only after the next request is actually received at time t = 4, if required. Relate this with Rule #3 mentioned in problem description.

Lift #2 will get unallocated at floor 5 at time t = (4 + 3) = 7.

Now, if Lift #1 is used to respond request #3, waiting time would be 4 seconds since Lift#1 is at floor #3 and will need 4 seconds to reach floor #7.

In this case, waiting time of all requests - {0,0,4} and the maximum waiting time is 4.

Instead, if Lift #2 is used to respond request #3, waiting time would be calculated as follows

Lift #2 will get unallocated at time t = 7, so we will have to wait 7-6 = 1 seconds for lift #2 to complete it's previous request.

Time needed for Lift #2 to travel from floor #5 to floor #7 = 7-5 = 2 seconds. Therefore, total waiting time = 1 + 2 = 3 seconds.

In this case, waiting time of all requests - {0, 0, 3} and the maximum waiting time is 3.

As we have to minimize the maximum waiting time, since 2nd option is yielding lesser waiting time than 1st option, 2nd option is the answer.

Therefore, the output is 3.

Example 2

Input

5 3

```
0 2 3

4 2 5
```

6 7 3

3 5 6

2 5 7

Output

1

Explanation

There are 5 requests and 3 lifts.

Initially, at t = 0, lift #1 will be at floor #2 whereas lift #2 and #3 will be at floor#5 (Initial allocation).

Request #1, #4, #5 can be responded instantly, i.e. waiting time = 0.

Lift #1 will get unallocated at floor 3 at time t = 1.

Lift #2 will get unallocated at floor 7 at time t = 2 + 2 = 4.

Lift #3 will get unallocated at floor 6 at time t = 3 + 1 = 4.

Request #2 can be served by lift #1. Waiting time would be 2 - 1 = 1.

Now, lift #1 will get unallocated at floor #5 at time t = 4 + 1 + 3 = 8.

Request #3 can be served by lift #3 as it will be floor #6 at t = 4. Waiting time = 0.

Waiting time of all requests - {0, 1, 0, 0, 0}.

Therefore, the output is 1.

##
##################################

Paste Reduction

Problem

Description

Some keys in Codu's computer keyboard are not working. Fortunately for Codu, these characters are present in a previously existing text file.

He wants to write a paragraph which involves typing those characters whose keys are defunct in Codu's keyboard. Only option left for Codu is to copy-paste those characters from the previously existing text files. However, copy-pasting keys is a laborious operation since one has to switch windows and also previously copied items are lost once a new set of characters are copied. Hence, Codu wants to minimize the number of times he needs to copy-paste from that text file. Fortunately there can be situations where previously copied characters are readily available for pasting. Help Codu devise a
method to minimize the number of times a paste operation is needed, given - the text he intends to type and the faulty keys

Constraints

0 < Length of paragraph <= 1000 characters

0 < number of faulty keys in keyboard <= 36

Only a to z and 0 to 9 keys can be faulty.

Input paragraph will not contain any upper-case letter.

Input

First line contains a paragraph that is be to be written.

Second line contains a string. This string has to be interpreted in the following fashion

All characters in that string correspond to faulty keys

That string is available for copy as-is from the other text file that Codu is referring

Also, individual characters can always be copied from the same text file

Output

Single integer denoting minimum number of copy operations required

Time

Limit 1

Exampl

es

Exampl

e 1

Input

supreme court is the highest judicial court su

Output

4

Explanation

Codu will first paste su from the file when typing characters su in the word supreme.

In the second instance, Codu will need to copy character u and paste it when typing character u in the word court.

In the third instance, Codu will need to copy character su and paste su when typing character s in the word is. Codu will back track using the left arrow key and type the characters "the highe".

In the fourth instance, Codu will again paste su. The overall string typed until this moment is "supreme court is the highesuu". Codu will then back track again using left arrow key and type characters "t j". At this point the typed string is "supreme court is the highest juu". Cursor is after character j. Codu will use right arrow key and now the cursor will be after ju. Codu will now type "udicial co". String typed till this point is "supreme court is the highest judicial cou". Cursor is after character o. Codu will use right arrow and the cursor will be after character u. Finally Codu will type "rt".

Final string - "supreme court is the highest judicial court" - is thus fully typed. Here, the number of paste operations are 4.

###

Count Pairs

Problem Description

Given an array of integers A, and an integer K find number of happy elements.

Element X is happy if there exists at least 1 element whose difference is less than K i.e. an element X is happy, if there is another element in the range [X-K, X+K] other than X itself.

Constraints

1 <= N <= 10^5

0 <= K <= 10^5

0 <= A[i] <= 10^9

Input

First line contains two integers N and K where N is size of the array and K is a number as described above

Second line contains N integers separated by space.

Output

Print a single integer denoting the total number of happy elements.

Time

Limit 1

Exampl

es

Exampl

e 1

Input

6 3

5 5 7 9 15 2

Output

```
5
```

Explanation

Other than number 15, everyone has at least 1 element in the range [X-3, X+3]. Hence they are all happy elements. Since these five are in number, the output is 5.

Example 2

Input

3 2

1 3 5

Output

3

Explanation

All numbers have at least 1 element in the range [X-2, X+2]. Hence they are all happy elements. Since these three are in number, the output is 3.