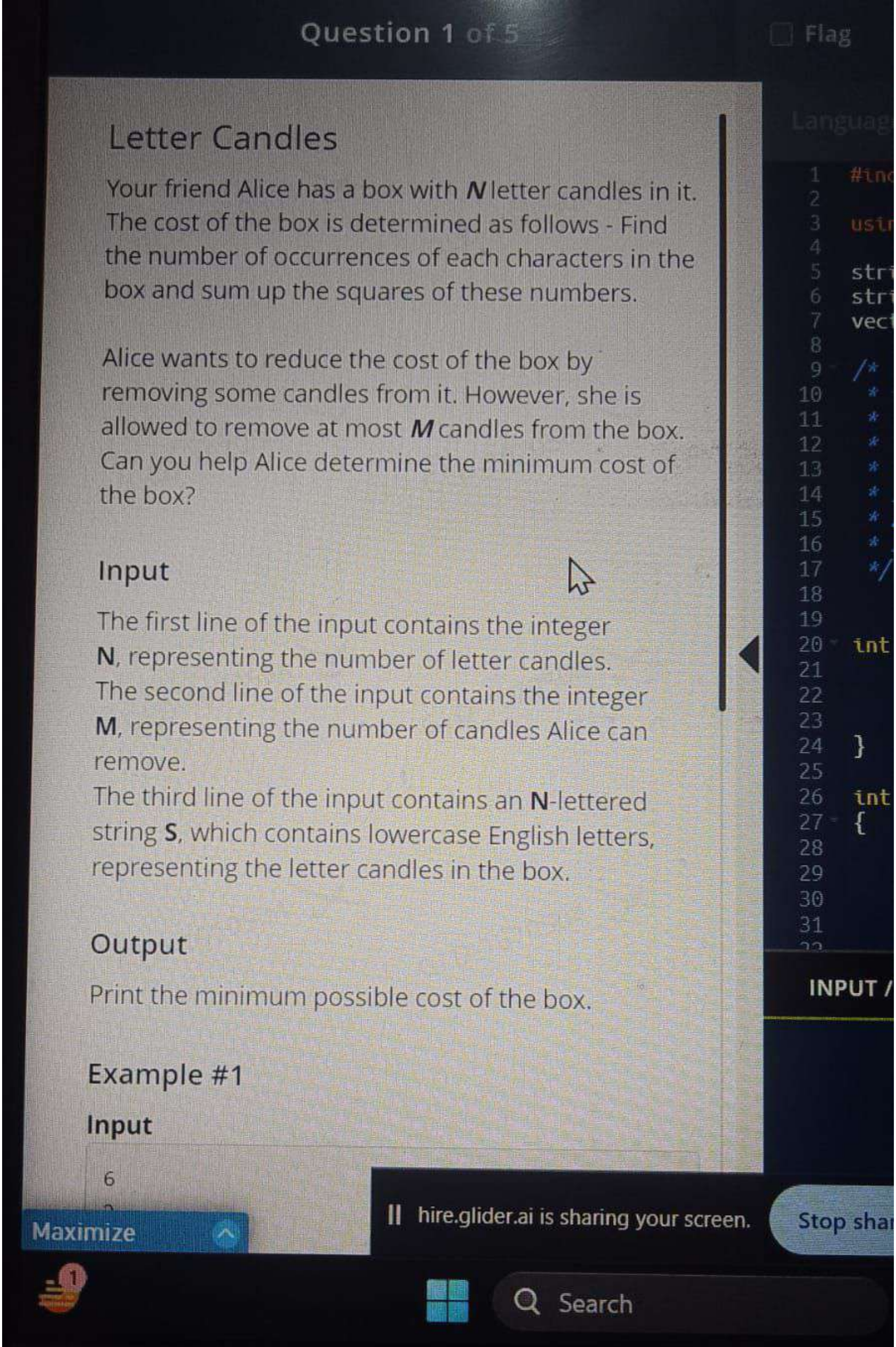


Question 1 of 5

□ Flag

# Letter Candles

Your friend Alice has a box with ***N*** letter candles in it. The cost of the box is determined as follows - Find the number of occurrences of each characters in the box and sum up the squares of these numbers.

Alice wants to reduce the cost of the box by removing some candles from it. However, she is allowed to remove at most ***M*** candles from the box. Can you help Alice determine the minimum cost of the box?

## Input

The first line of the input contains the integer **N**, representing the number of letter candles.
The second line of the input contains the integer **M**, representing the number of candles Alice can remove.
The third line of the input contains an **N**-lettered string **S**, which contains lowercase English letters, representing the letter candles in the box.

## Output

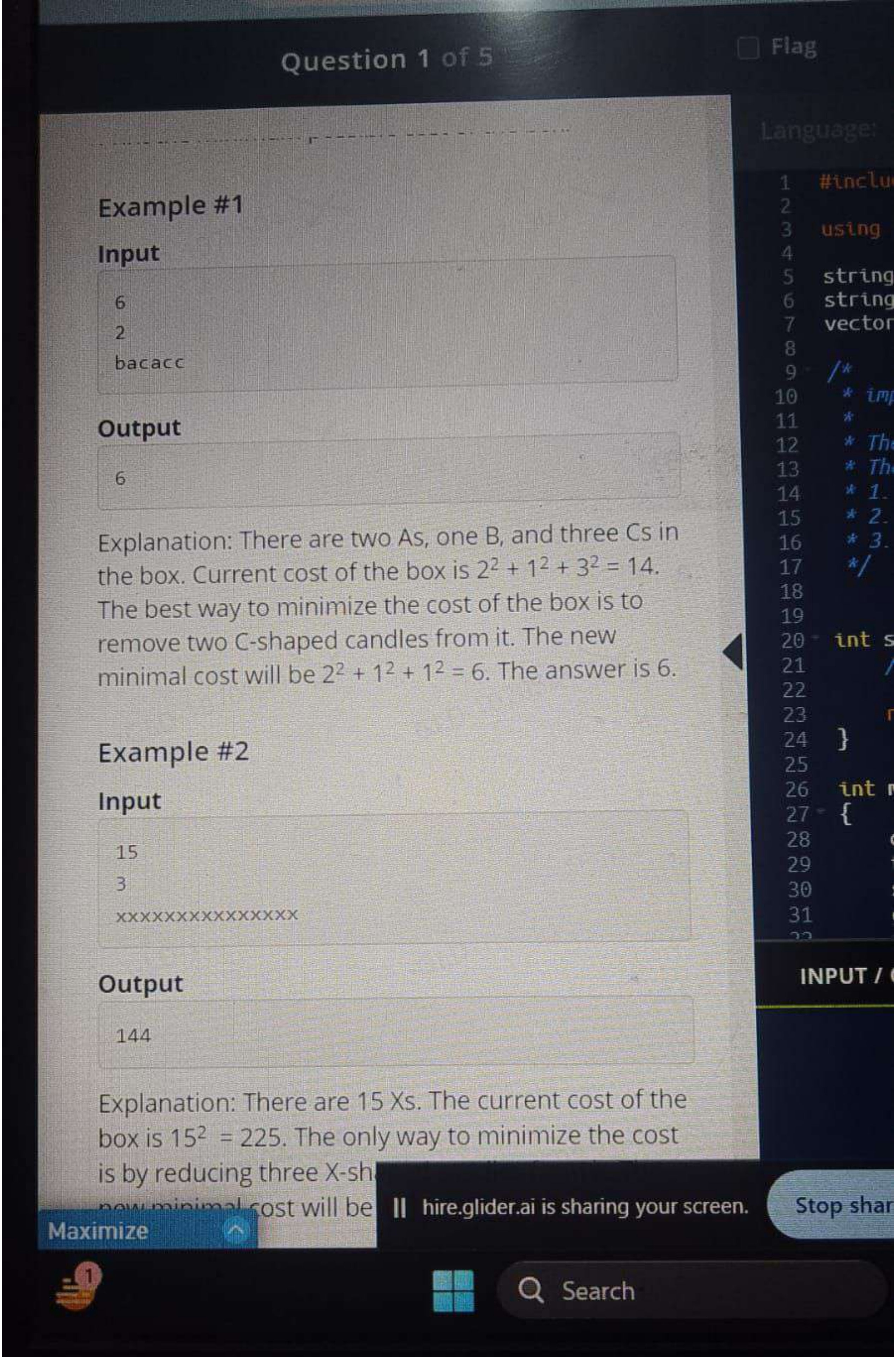
Print the minimum possible cost of the box.

## Example #1

### Input

```
6
2
```

```
#inc

usi

stri
stri
vect

/*
 *
 *
 *
 *
 *
 *
 *
 */


int



}

int
{
```



Question 1 of 5

### Example #1

**Input**

```
6
2
bacacc
```

**Output**

```
6
```

Explanation: There are two As, one B, and three Cs in the box. Current cost of the box is $2^2 + 1^2 + 3^2 = 14$. The best way to minimize the cost of the box is to remove two C-shaped candles from it. The new minimal cost will be $2^2 + 1^2 + 1^2 = 6$. The answer is 6.

### Example #2

**Input**

```
15
3
xxxxxxxxxxxxxxx
```

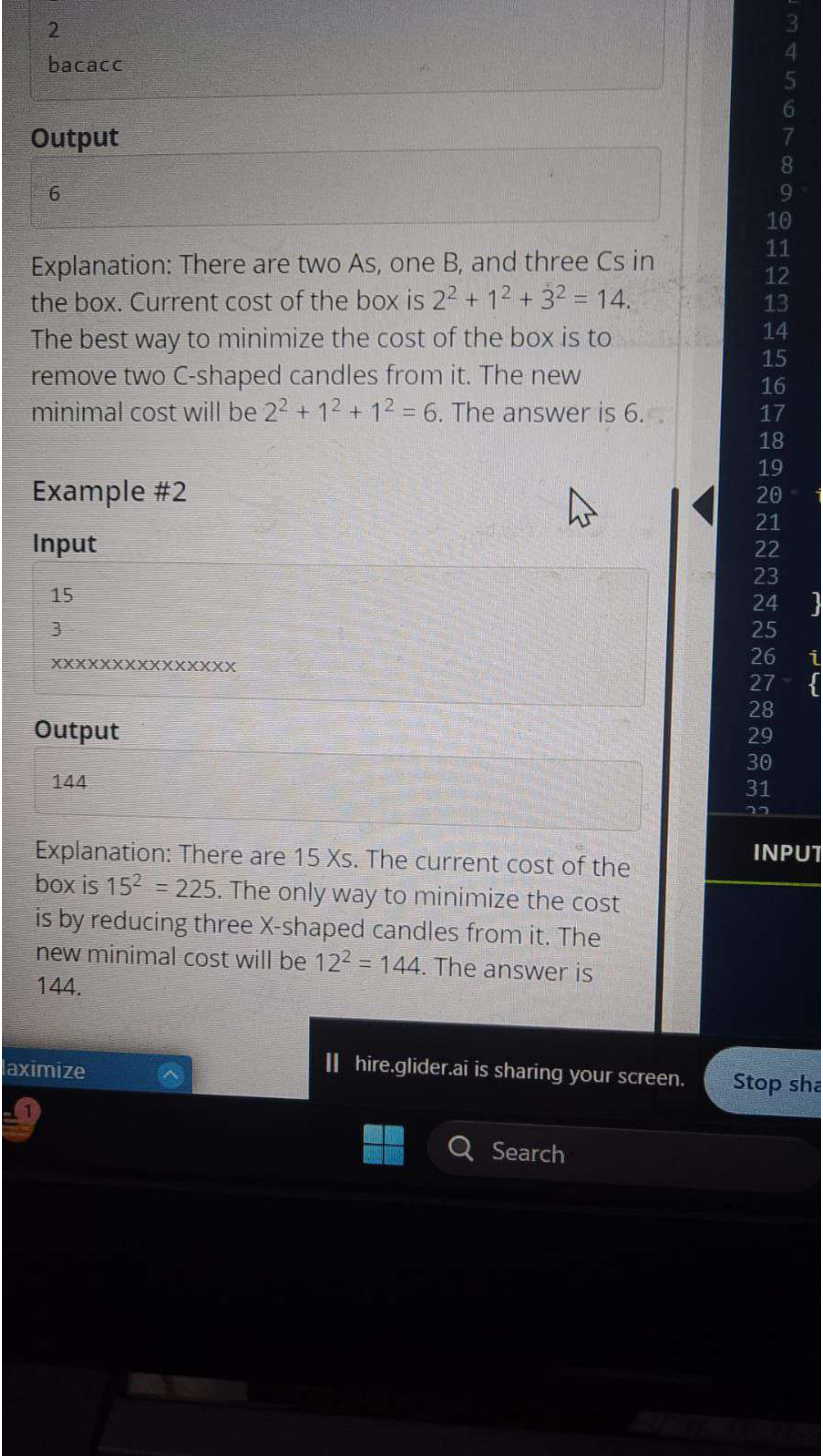
**Output**

```
144
```

Explanation: There are 15 Xs. The current cost of the box is $15^2 = 225$. The only way to minimize the cost is by reducing three X-sh

new minimal cost will be

```
#inclu

using

string
string
vector

/*
 * imp
 *
 * Th
 * Th
 * 1.
 * 2.
 * 3.
 */


int s
    /

    r
}

int r
{
```



```
2
bacacc
```

**Output**

```
6
```

Explanation: There are two As, one B, and three Cs in the box. Current cost of the box is $2^2 + 1^2 + 3^2 = 14$. The best way to minimize the cost of the box is to remove two C-shaped candles from it. The new minimal cost will be $2^2 + 1^2 + 1^2 = 6$. The answer is 6.

### Example #2

**Input**

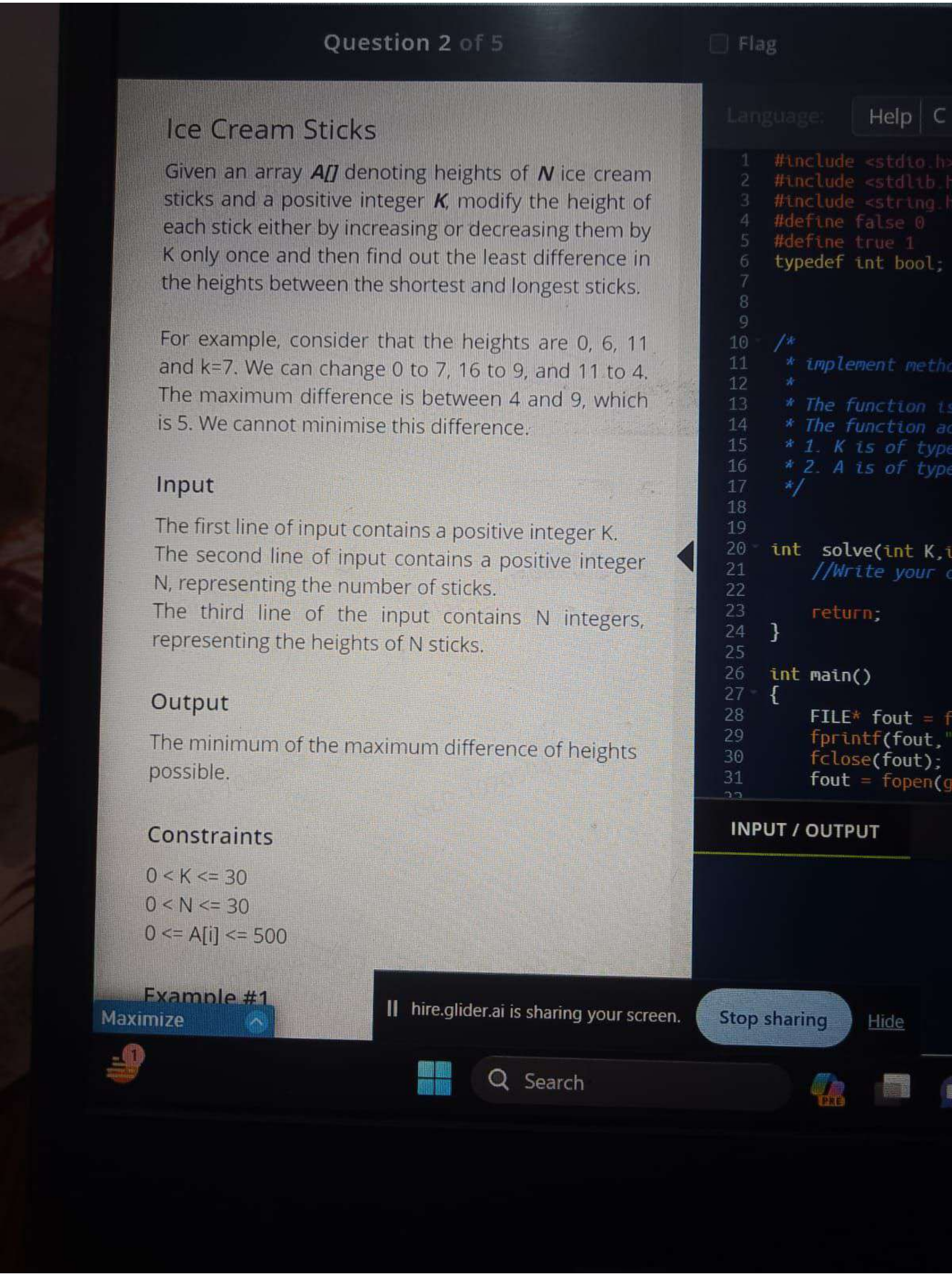
```
15
3
xxxxxxxxxxxxxxx
```

**Output**

```
144
```

Explanation: There are 15 Xs. The current cost of the box is $15^2 = 225$. The only way to minimize the cost is by reducing three X-shaped candles from it. The new minimal cost will be $12^2 = 144$. The answer is 144.

hire.glider.ai is sharing your screen.



## Ice Cream Sticks

Given an array *A[]* denoting heights of ***N*** ice cream sticks and a positive integer ***K***, modify the height of each stick either by increasing or decreasing them by K only once and then find out the least difference in the heights between the shortest and longest sticks.

For example, consider that the heights are 0, 6, 11 and k=7. We can change 0 to 7, 16 to 9, and 11 to 4. The maximum difference is between 4 and 9, which is 5. We cannot minimise this difference.

### Input

The first line of input contains a positive integer K.
The second line of input contains a positive integer N, representing the number of sticks.
The third line of the input contains N integers, representing the heights of N sticks.

### Output

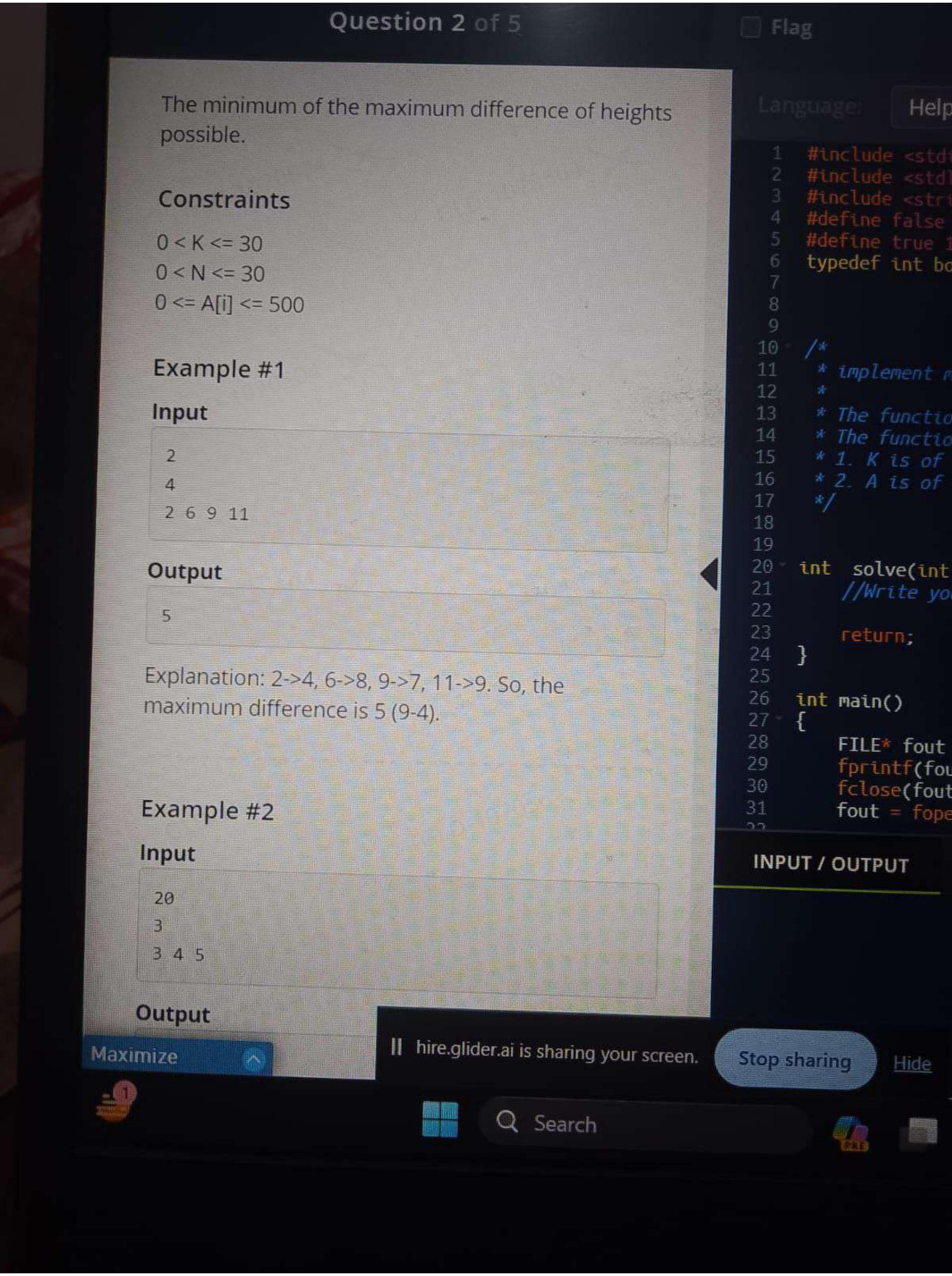
The minimum of the maximum difference of heights possible.

### Constraints

0 < K <= 30
0 < N <= 30
0 <= A[i] <= 500

### Example #1

```
#include <stdio.h>
#include <stdlib.h
#include <string.h
#define false 0
#define true 1
typedef int bool;



/*
 * implement metho
 *
 * The function is
 * The function ac
 * 1. K is of type
 * 2. A is of type
 */


int  solve(int K,i
    //Write your c

    return;
}

int main()
{
    FILE* fout = f
    fprintf(fout,"
    fclose(fout);
    fout = fopen(g
```



The minimum of the maximum difference of heights possible.

### Constraints

0 < K <= 30
0 < N <= 30
0 <= A[i] <= 500

### Example #1

**Input**

```
2
4
2 6 9 11
```

**Output**

```
5
```

Explanation: 2->4, 6->8, 9->7, 11->9. So, the maximum difference is 5 (9-4).

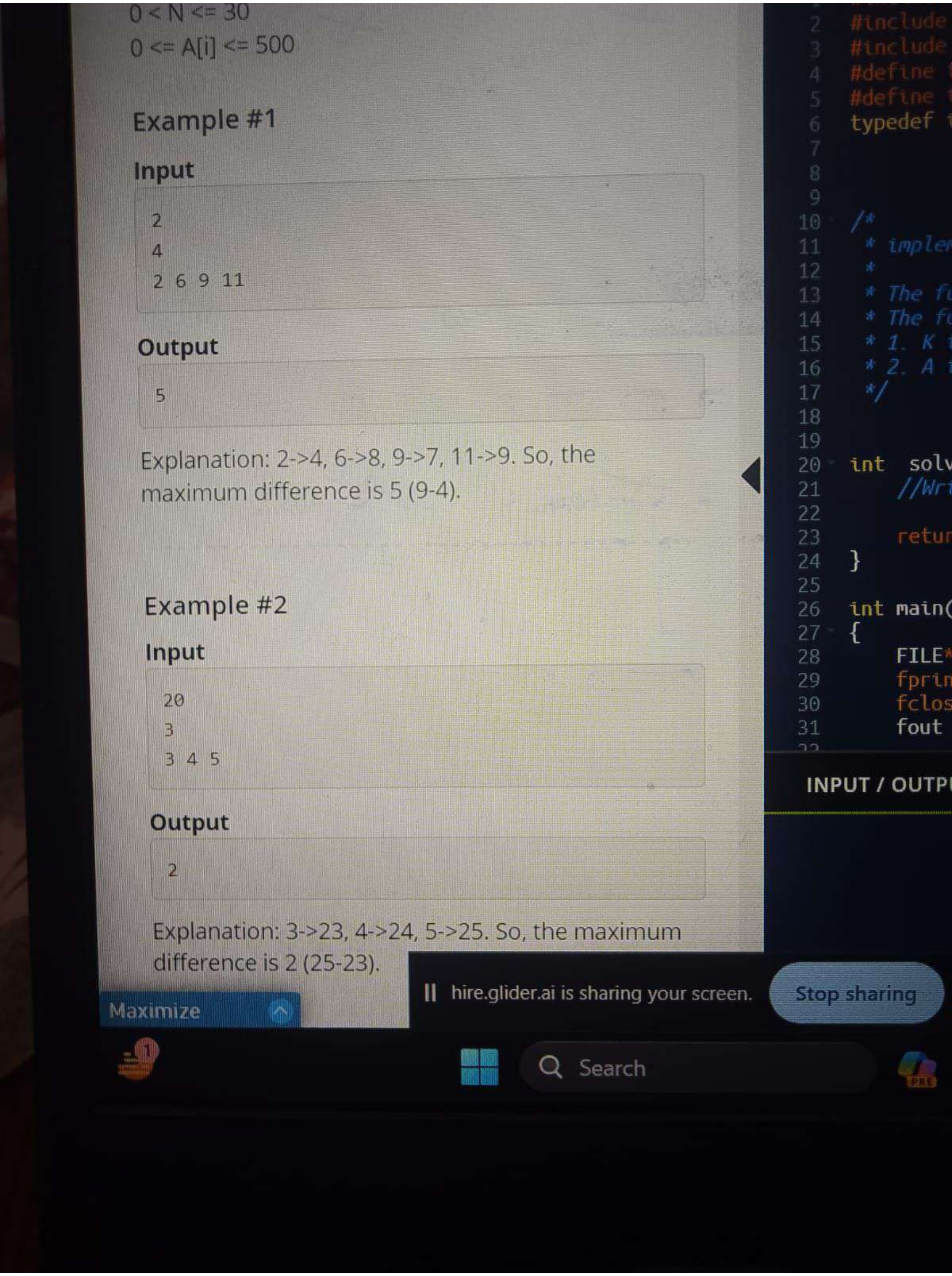
### Example #2

**Input**

```
20
3
3 4 5
```

**Output**

```
#include <stdi
#include <stdl
#include <stri
#define false
#define true 1
typedef int bo



/*
 * implement m
 *
 * The functio
 * The functio
 * 1. K is of
 * 2. A is of
 */


int  solve(int
    //Write yo

    return;
}

int main()
{
    FILE* fout
    fprintf(fou
    fclose(fout
    fout = fope
```



0 < N <= 30

0 <= A[i] <= 500

## Example #1

**Input**

```
2
4
2 6 9 11
```

**Output**

```
5
```

Explanation: 2->4, 6->8, 9->7, 11->9. So, the maximum difference is 5 (9-4).

## Example #2

**Input**

```
20
3
3 4 5
```

**Output**

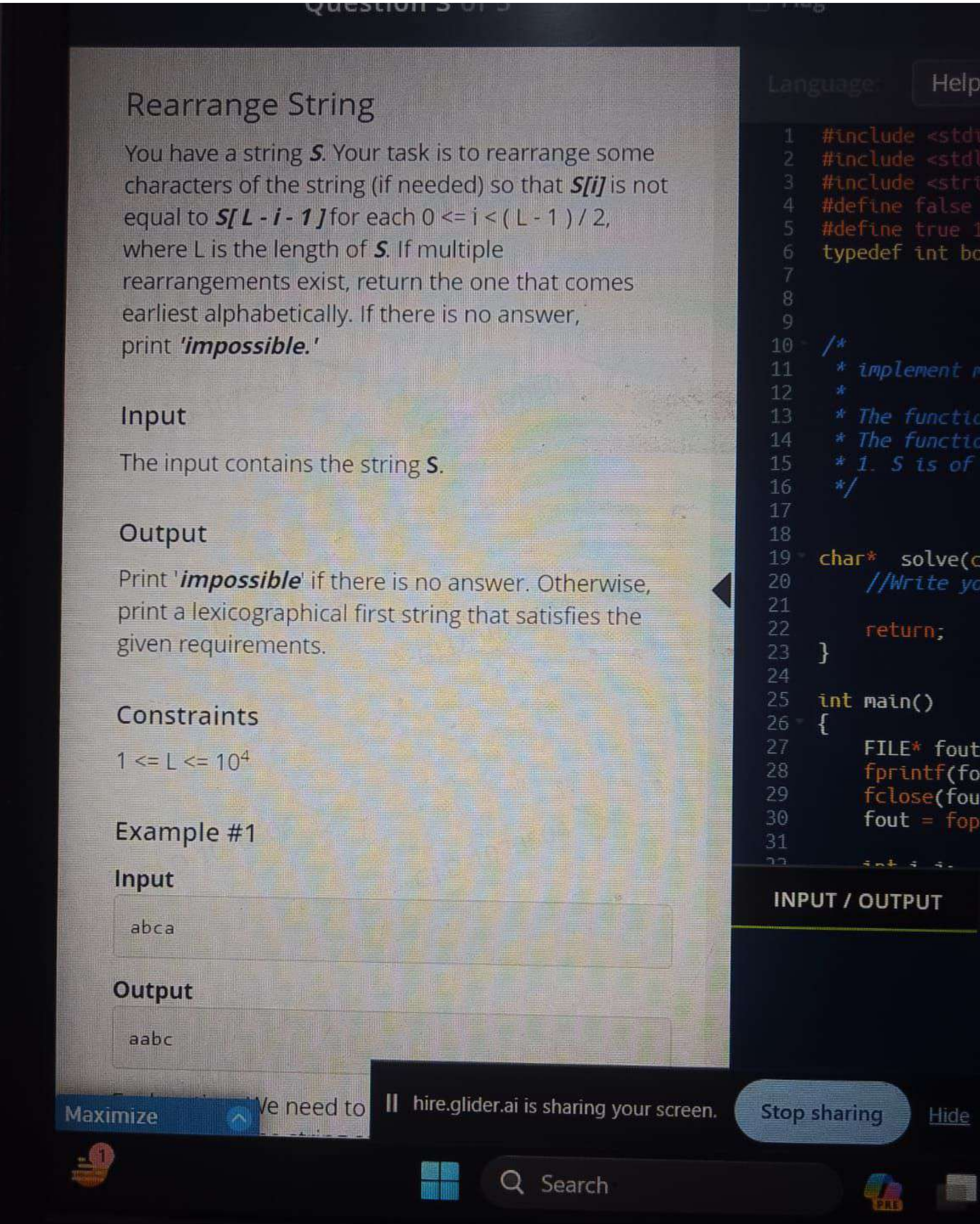
```
2
```

Explanation: 3->23, 4->24, 5->25. So, the maximum difference is 2 (25-23).

```
#include
#include
#define
#define
typedef



/*
 * implem
 *
 * The fu
 * The fu
 * 1. K
 * 2. A
 */


int  solv
    //Wri

    retur
}

int main(
{
    FILE*
    fprin
    fclos
    fout
```

INPUT / OUTPU

hire.glider.ai is sharing your screen.



## Rearrange String

You have a string ***S***. Your task is to rearrange some characters of the string (if needed) so that ***S[i]*** is not equal to ***S[ L - i - 1 ]*** for each 0 <= i < ( L - 1 ) / 2, where L is the length of ***S***. If multiple rearrangements exist, return the one that comes earliest alphabetically. If there is no answer, print ***'impossible.'***

### Input

The input contains the string **S**.

### Output

Print '***impossible***' if there is no answer. Otherwise, print a lexicographical first string that satisfies the given requirements.

### Constraints

$1 <= L <= 10^4$

### Example #1

**Input**

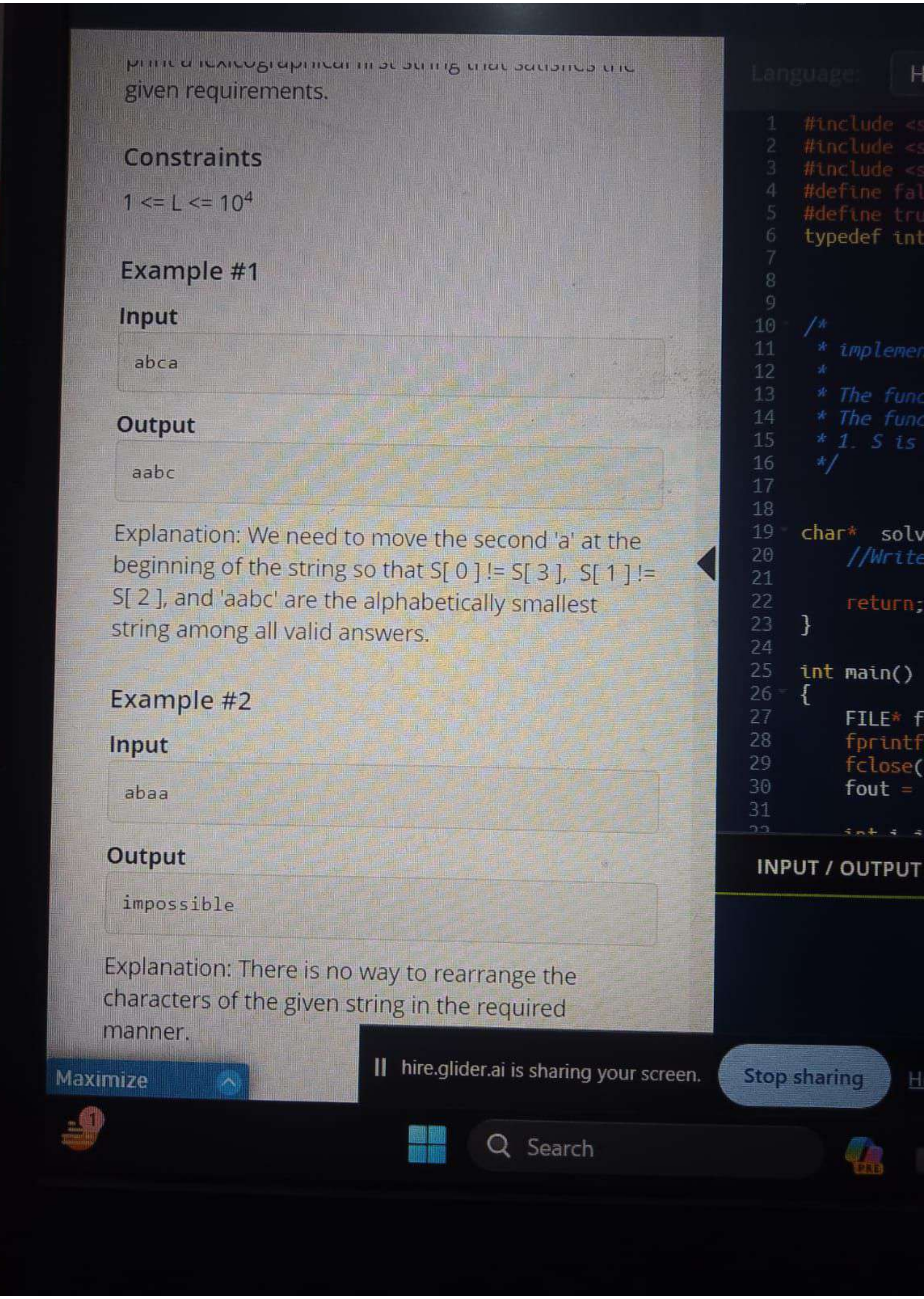
```
abca
```

**Output**

```
aabc
```

We need to

```
#include <stdi
#include <stdl
#include <stri
#define false
#define true 1
typedef int bo



/*
 * implement r
 *
 * The functio
 * The functio
 * 1. S is of
 */


char*  solve(c
    //Write yo

    return;
}

int main()
{
    FILE* fout
    fprintf(fo
    fclose(fou
    fout = fop
```



print a lexicographical first string that satisfies the given requirements.

### Constraints

1 <= L <= $10^4$

### Example #1

**Input**

```
abca
```

**Output**

```
aabc
```

Explanation: We need to move the second 'a' at the beginning of the string so that S[ 0 ] != S[ 3 ], S[ 1 ] != S[ 2 ], and 'aabc' are the alphabetically smallest string among all valid answers.

### Example #2

**Input**

```
abaa
```

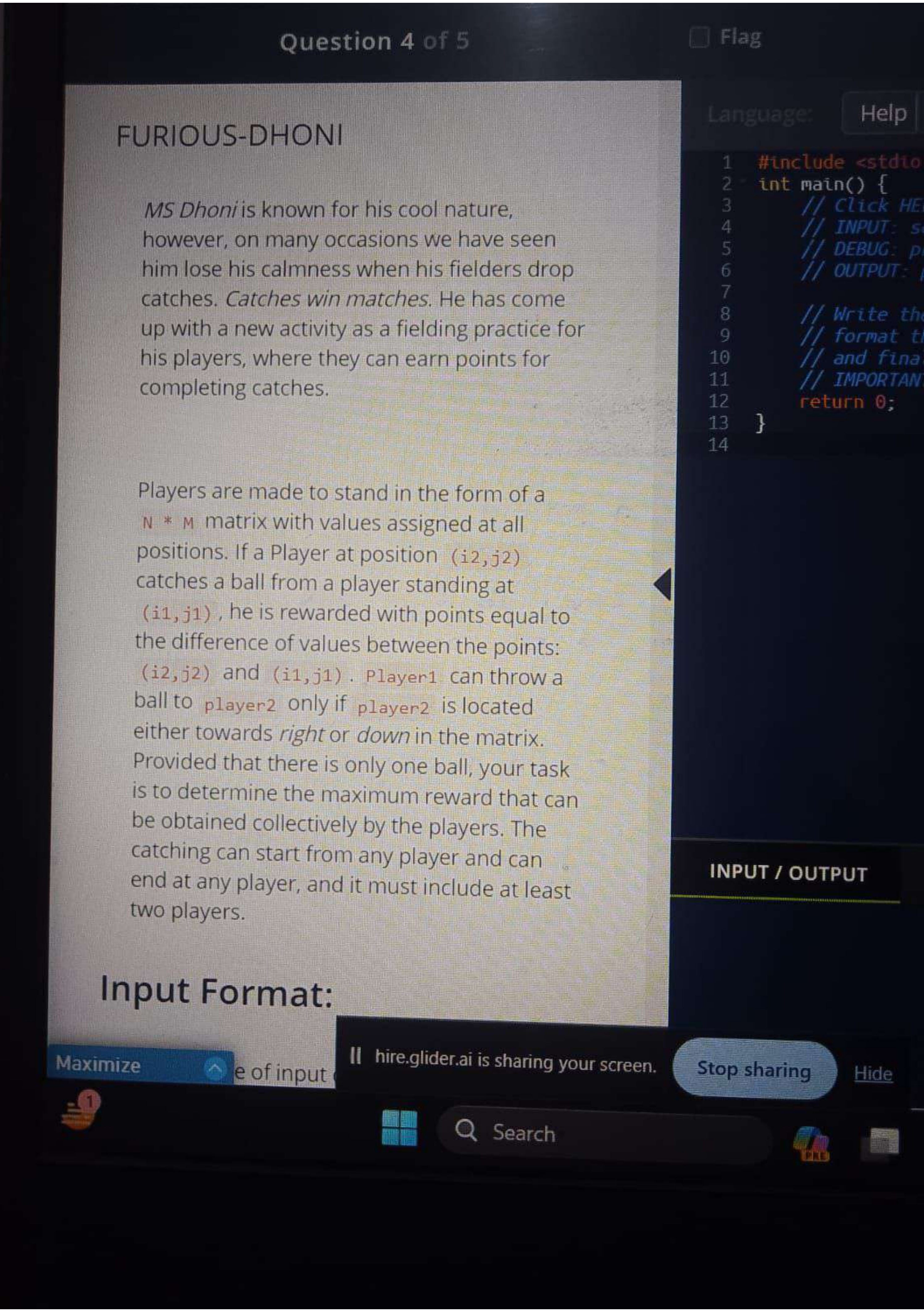
**Output**

```
impossible
```

Explanation: There is no way to rearrange the characters of the given string in the required manner.

Language: H

```
#include <s
#include <s
#include <s
#define fal
#define tru
typedef int



/*
 * implemen
 *
 * The func
 * The func
 * 1. S is
 */


char*  solv
    //Write

    return;
}

int main()
{
    FILE* f
    fprintf
    fclose(
    fout =

```



## FURIOUS-DHONI

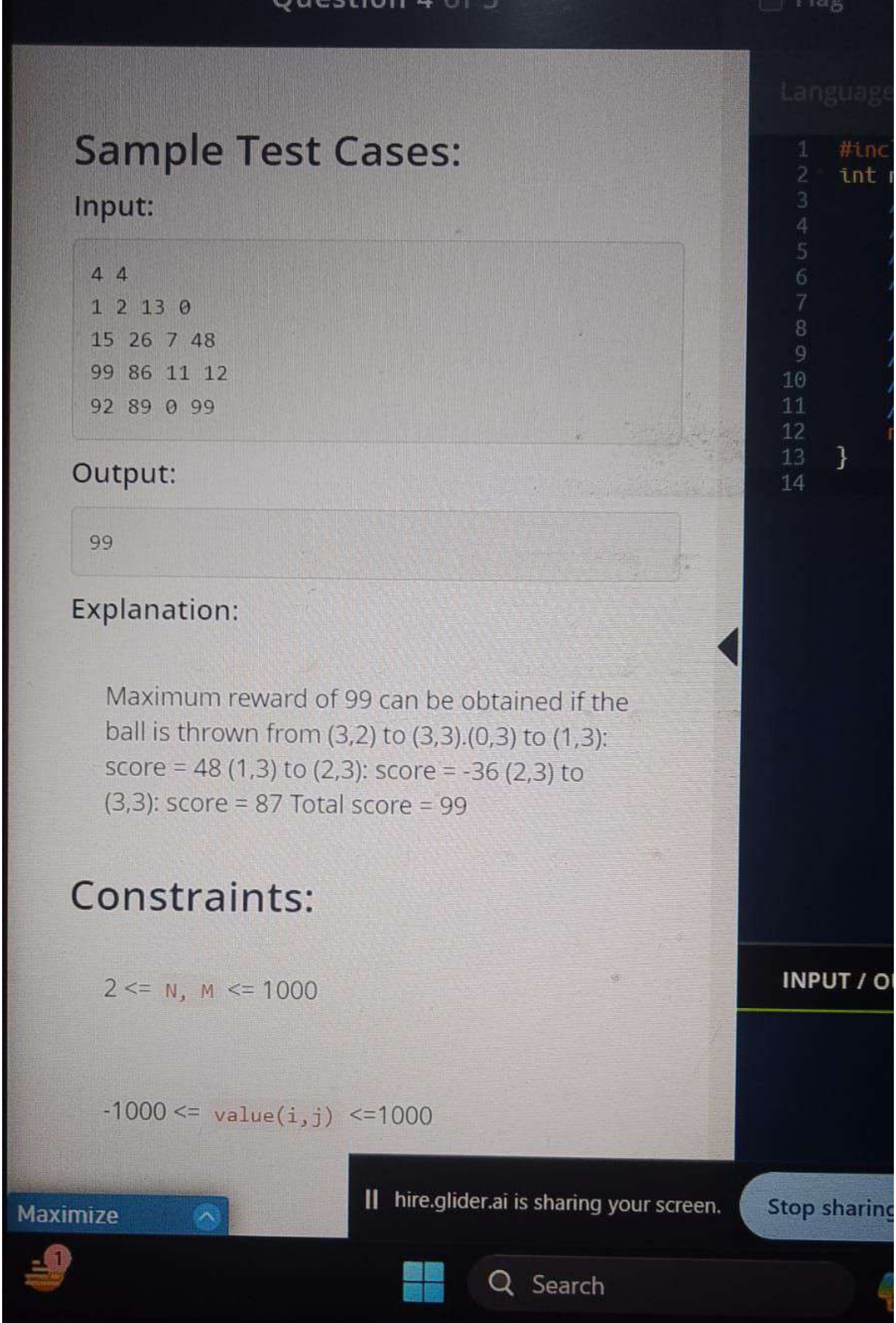
*MS Dhoni* is known for his cool nature, however, on many occasions we have seen him lose his calmness when his fielders drop catches. *Catches win matches.* He has come up with a new activity as a fielding practice for his players, where they can earn points for completing catches.

Players are made to stand in the form of a `N * M` matrix with values assigned at all positions. If a Player at position `(i2,j2)` catches a ball from a player standing at `(i1,j1)`, he is rewarded with points equal to the difference of values between the points: `(i2,j2)` and `(i1,j1)`. `Player1` can throw a ball to `player2` only if `player2` is located either towards *right* or *down* in the matrix. Provided that there is only one ball, your task is to determine the maximum reward that can be obtained collectively by the players. The catching can start from any player and can end at any player, and it must include at least two players.

## Input Format:

e of input

```
#include <stdio
int main() {
    // Click HE
    // INPUT: s
    // DEBUG: p
    // OUTPUT: 

    // Write th
    // format t
    // and fina
    // IMPORTAN
    return 0;
}
```



```
#inc
int r


}
```

## Sample Test Cases:

Input:

```
4 4
1 2 13 0
15 26 7 48
99 86 11 12
92 89 0 99
```

Output:

```
99
```

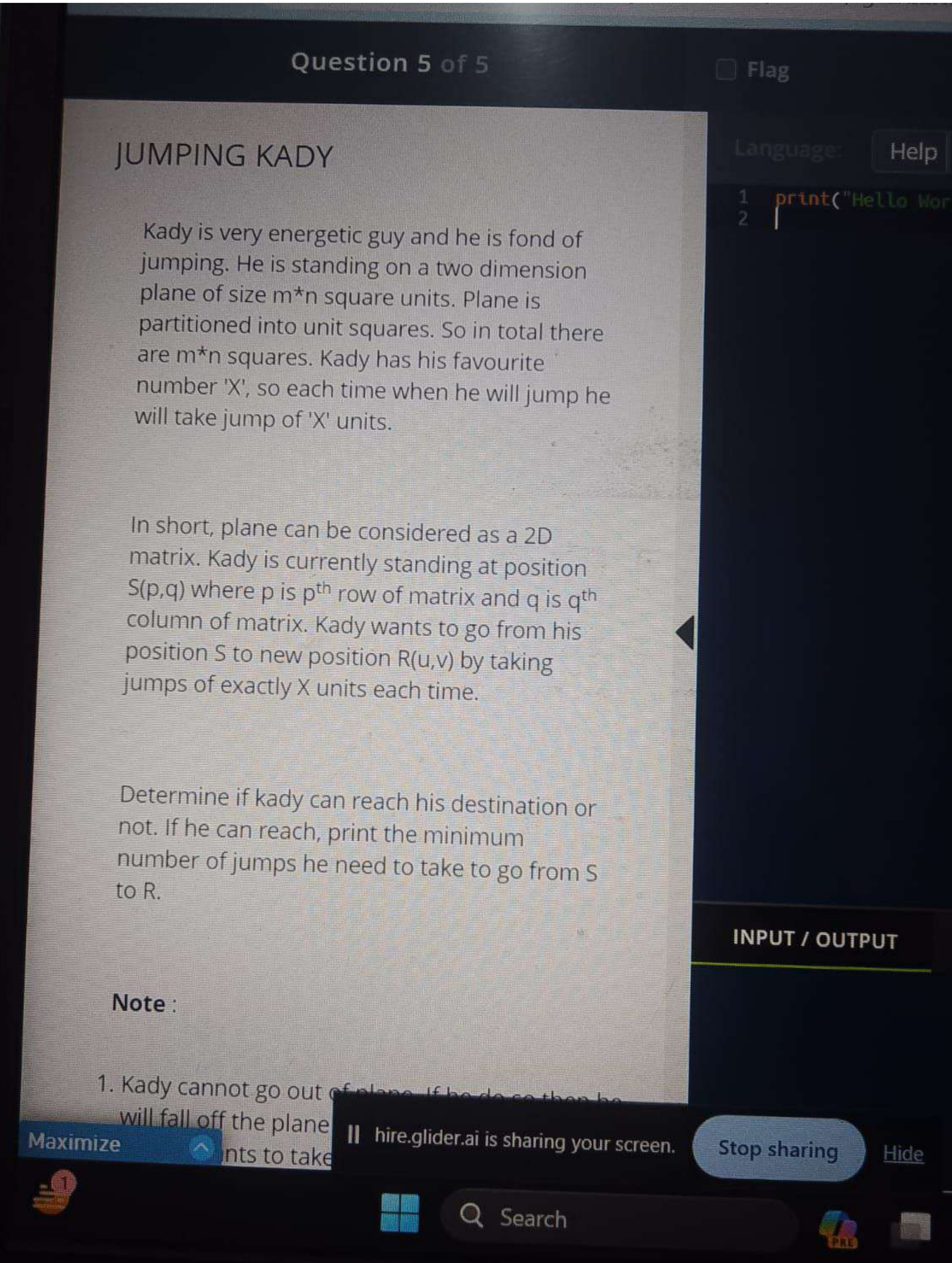
Explanation:

Maximum reward of 99 can be obtained if the ball is thrown from (3,2) to (3,3).(0,3) to (1,3): score = 48 (1,3) to (2,3): score = -36 (2,3) to (3,3): score = 87 Total score = 99

## Constraints:

2 <= `N, M` <= 1000

-1000 <= `value(i,j)` <=1000



# JUMPING KADY

Kady is very energetic guy and he is fond of jumping. He is standing on a two dimension plane of size m*n square units. Plane is partitioned into unit squares. So in total there are m*n squares. Kady has his favourite number 'X', so each time when he will jump he will take jump of 'X' units.

In short, plane can be considered as a 2D matrix. Kady is currently standing at position S(p,q) where p is $p^{th}$ row of matrix and q is $q^{th}$ column of matrix. Kady wants to go from his position S to new position R(u,v) by taking jumps of exactly X units each time.

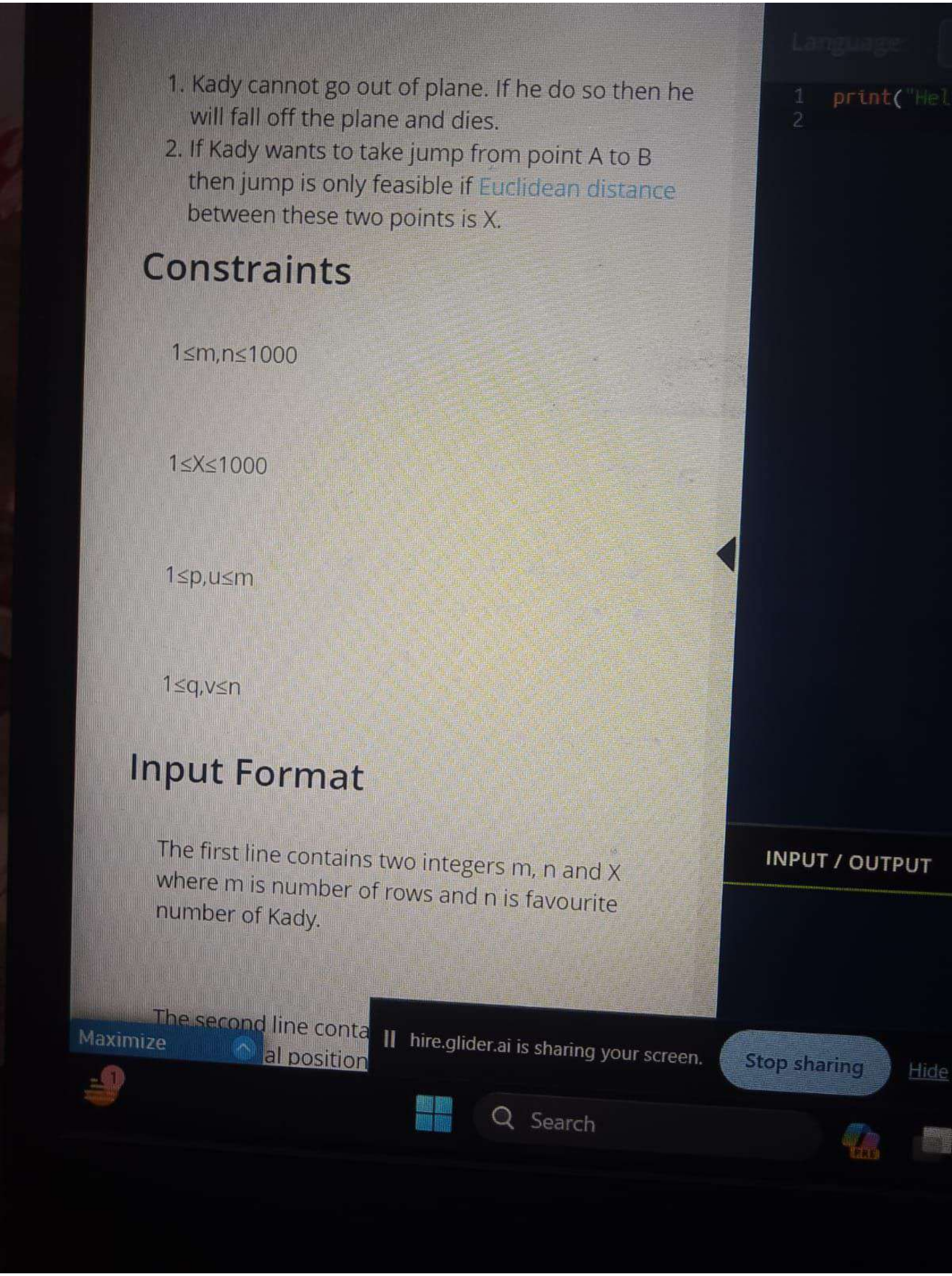
Determine if kady can reach his destination or not. If he can reach, print the minimum number of jumps he need to take to go from S to R.

**Note** :

1. Kady cannot go out of plane. If he do so then he will fall off the plane
nts to take

```
print("Hello Wor
```

hire.glider.ai is sharing your screen.



1. Kady cannot go out of plane. If he do so then he will fall off the plane and dies.
2. If Kady wants to take jump from point A to B then jump is only feasible if Euclidean distance between these two points is X.

## Constraints

1≤m,n≤1000

1≤X≤1000

1≤p,u≤m

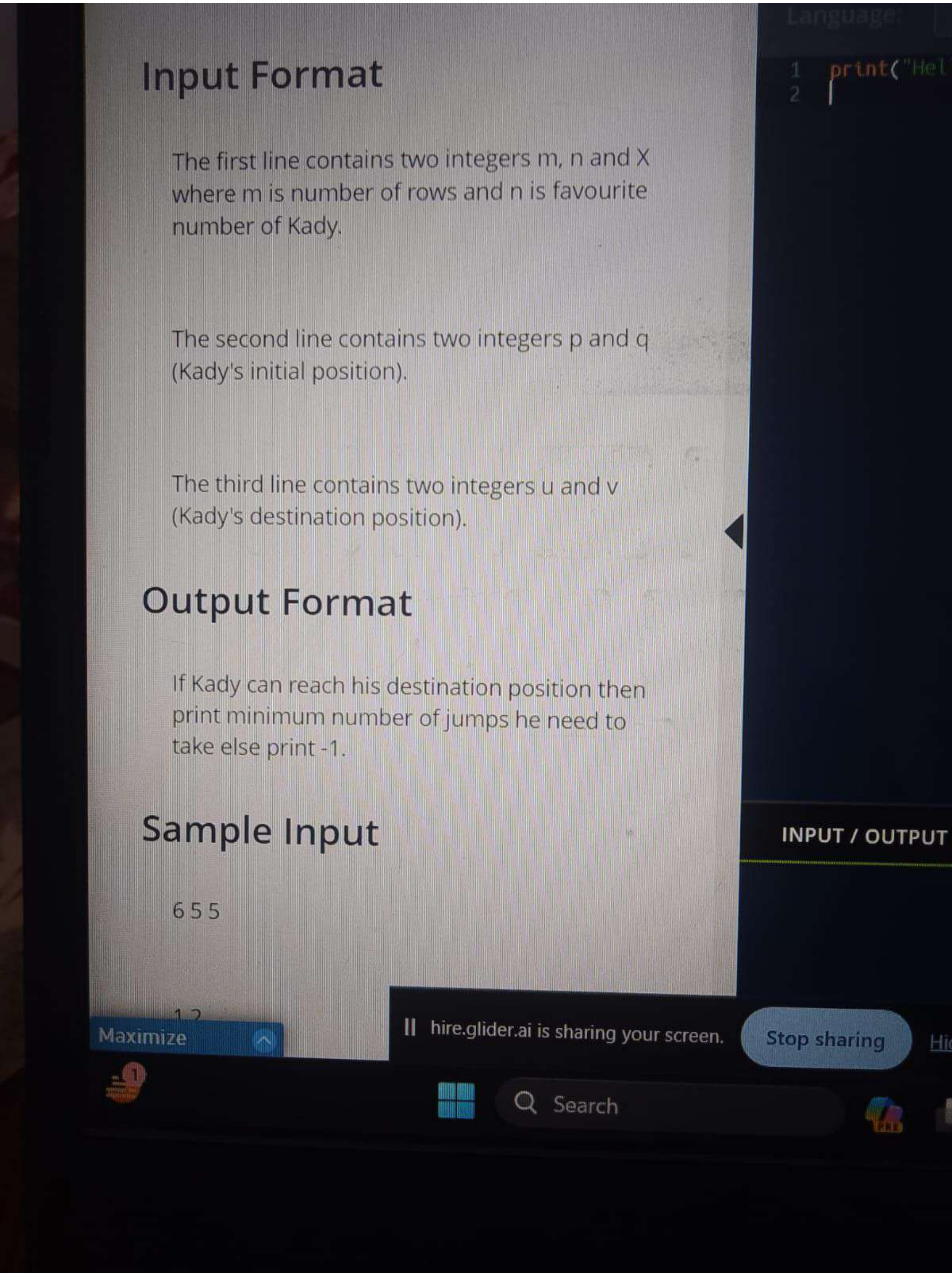
1≤q,v≤n

## Input Format

The first line contains two integers m, n and X where m is number of rows and n is favourite number of Kady.

The second line conta
al position

```
print("Hel
```



## Input Format

The first line contains two integers m, n and X where m is number of rows and n is favourite number of Kady.

The second line contains two integers p and q (Kady's initial position).

The third line contains two integers u and v (Kady's destination position).

## Output Format

If Kady can reach his destination position then print minimum number of jumps he need to take else print -1.

## Sample Input

6 5 5

1 2

```
print("Hel
```