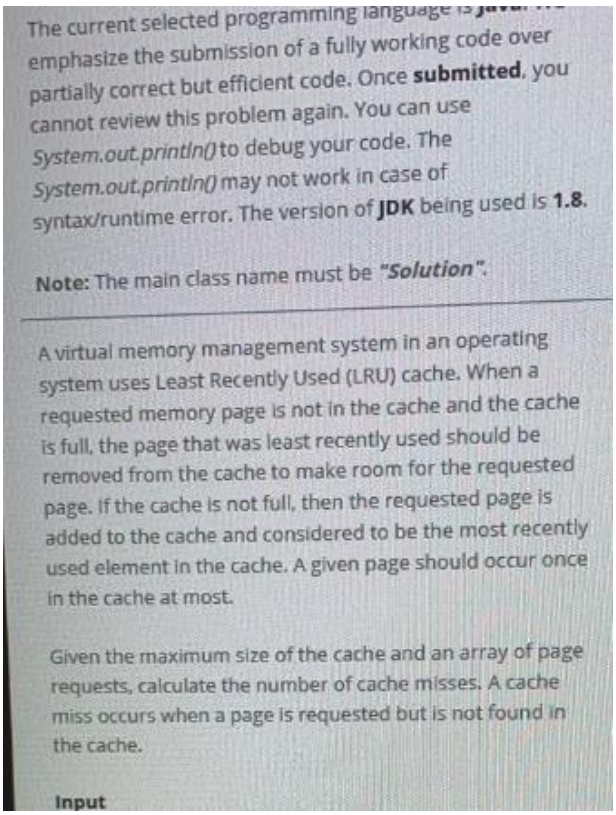
Problem-13: A virtual memory management system in an operating system uses Least Recently Used (LRU) cache.



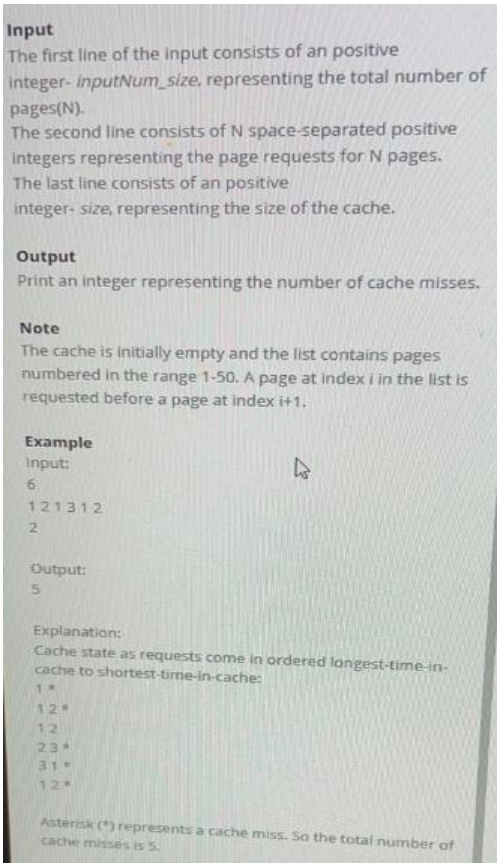
The current selected programming language is **Java**. Please emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *System.out.println()* to debug your code. The *System.out.println()* may not work in case of syntax/runtime error. The version of **JDK** being used is **1.8.**

Note: The main class name must be ***"Solution"***.

A virtual memory management system in an operating system uses Least Recently Used (LRU) cache. When a requested memory page is not in the cache and the cache is full, the page that was least recently used should be removed from the cache to make room for the requested page. If the cache is not full, then the requested page is added to the cache and considered to be the most recently used element in the cache. A given page should occur once in the cache at most.

Given the maximum size of the cache and an array of page requests, calculate the number of cache misses. A cache miss occurs when a page is requested but is not found in the cache.

Input



Input
The first line of the input consists of an positive integer- *inputNum_size*, representing the total number of pages(N).
The second line consists of N space-separated positive integers representing the page requests for N pages.
The last line consists of an positive integer- *size*, representing the size of the cache.

Output
Print an integer representing the number of cache misses.

Note
The cache is initially empty and the list contains pages numbered in the range 1-50. A page at index *i* in the list is requested before a page at index i+1.

Example
Input:
6
1 2 1 3 1 2
2

Output:
5

Explanation:
Cache state as requests come in ordered longest-time-in-cache to shortest-time-in-cache:
1 *
1 2 *
1 2
2 3 *
3 1 *
1 2 *

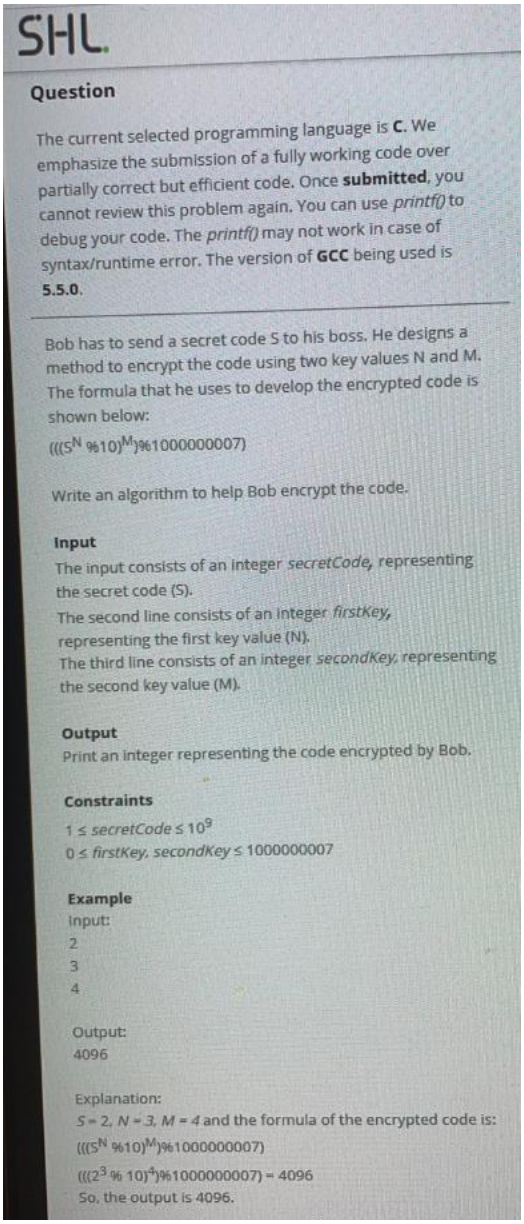
Asterisk (*) represents a cache miss. So the total number of cache misses is 5.

Consider the priority_queue<pair<int, int>> (descending order of age_bit) with <age_bit, page_no>, age_bit will be set according to operations (starting with 1 initially) and will not be updated if cache-hit is found.
Also, maintain, is_present_in_cache Boolean array, to keep track of page present in cache or not, because there is not O(1) peep operation in priority queue
When cache is full will remove the entry from map that has age_bit lowest!

In real LRU, you should also consider when hit is found and update the age_bit by +1

Youtube Link (OG question): Click here.

Problem-0: Bob has to send a secret code S to his boss. He designs a method to encrypt



SHL.

Question

The current selected programming language is **C**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *printf()* to debug your code. The *printf()* may not work in case of syntax/runtime error. The version of **GCC** being used is **5.5.0**.

Bob has to send a secret code S to his boss. He designs a method to encrypt the code using two key values N and M. The formula that he uses to develop the encrypted code is shown below:

$(((S^N \%10)^M)\%1000000007)$

Write an algorithm to help Bob encrypt the code.

Input

The input consists of an integer *secretCode*, representing the secret code (S).
The second line consists of an integer *firstKey*, representing the first key value (N).
The third line consists of an integer *secondKey*, representing the second key value (M).

Output

Print an integer representing the code encrypted by Bob.

Constraints

$1 \le secretCode \le 10^9$
$0 \le firstKey, secondKey \le 1000000007$

Example

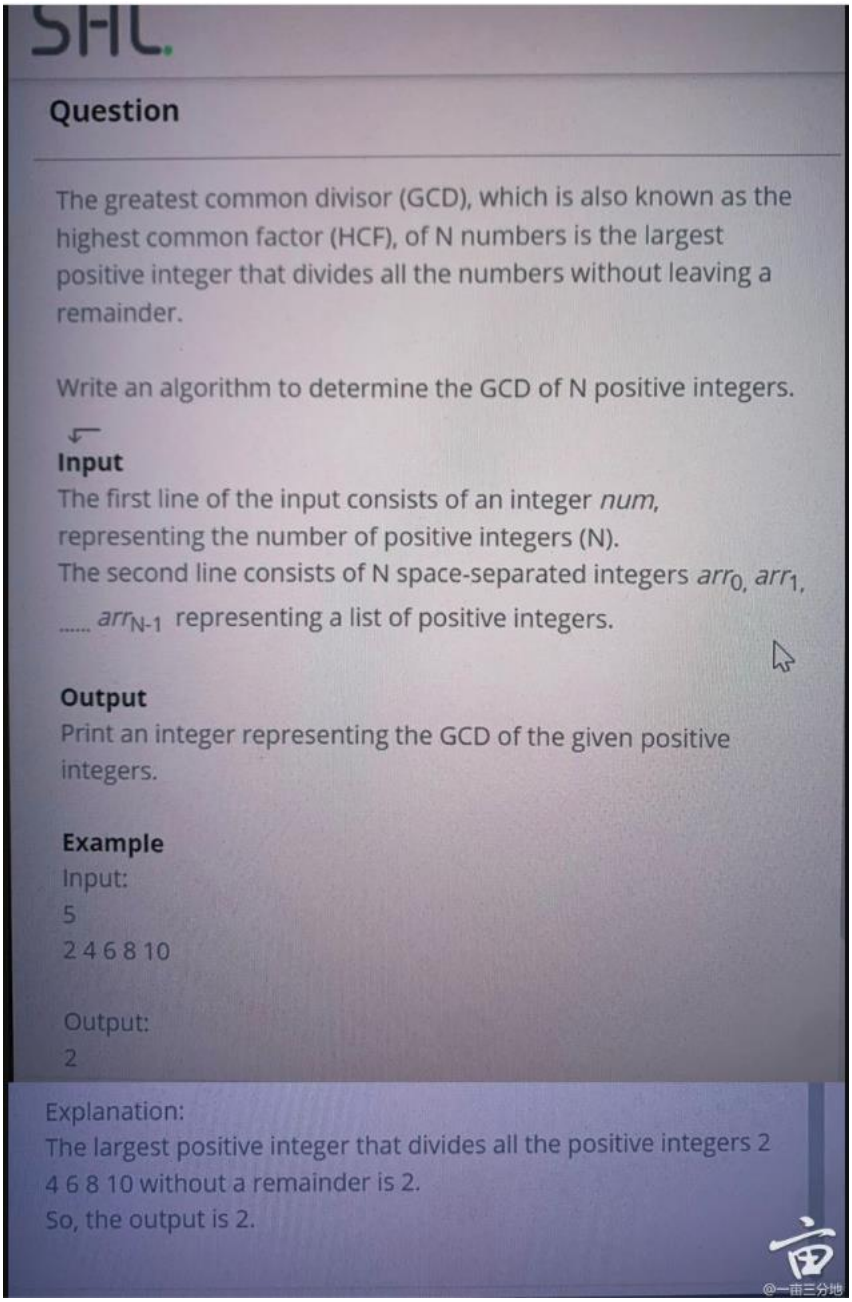
Input:
2
3
4

Output:
4096

Explanation:
$S = 2$, $N = 3$, $M = 4$ and the formula of the encrypted code is:
$(((S^N \%10)^M)\%1000000007)$
$(((2^3 \% 10)^4)\%1000000007) = 4096$
So, the output is 4096.

```
long long binpow(long long a, long long b, long long m) {
    a %= m;
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
```

Problem-1: GCD of N positive integers.



SHL.

Question

The greatest common divisor (GCD), which is also known as the highest common factor (HCF), of N numbers is the largest positive integer that divides all the numbers without leaving a remainder.

Write an algorithm to determine the GCD of N positive integers.

Input

The first line of the input consists of an integer *num*, representing the number of positive integers (N).
The second line consists of N space-separated integers arr_0, arr_1, ... arr_{N-1} representing a list of positive integers.

Output

Print an integer representing the GCD of the given positive integers.

Example

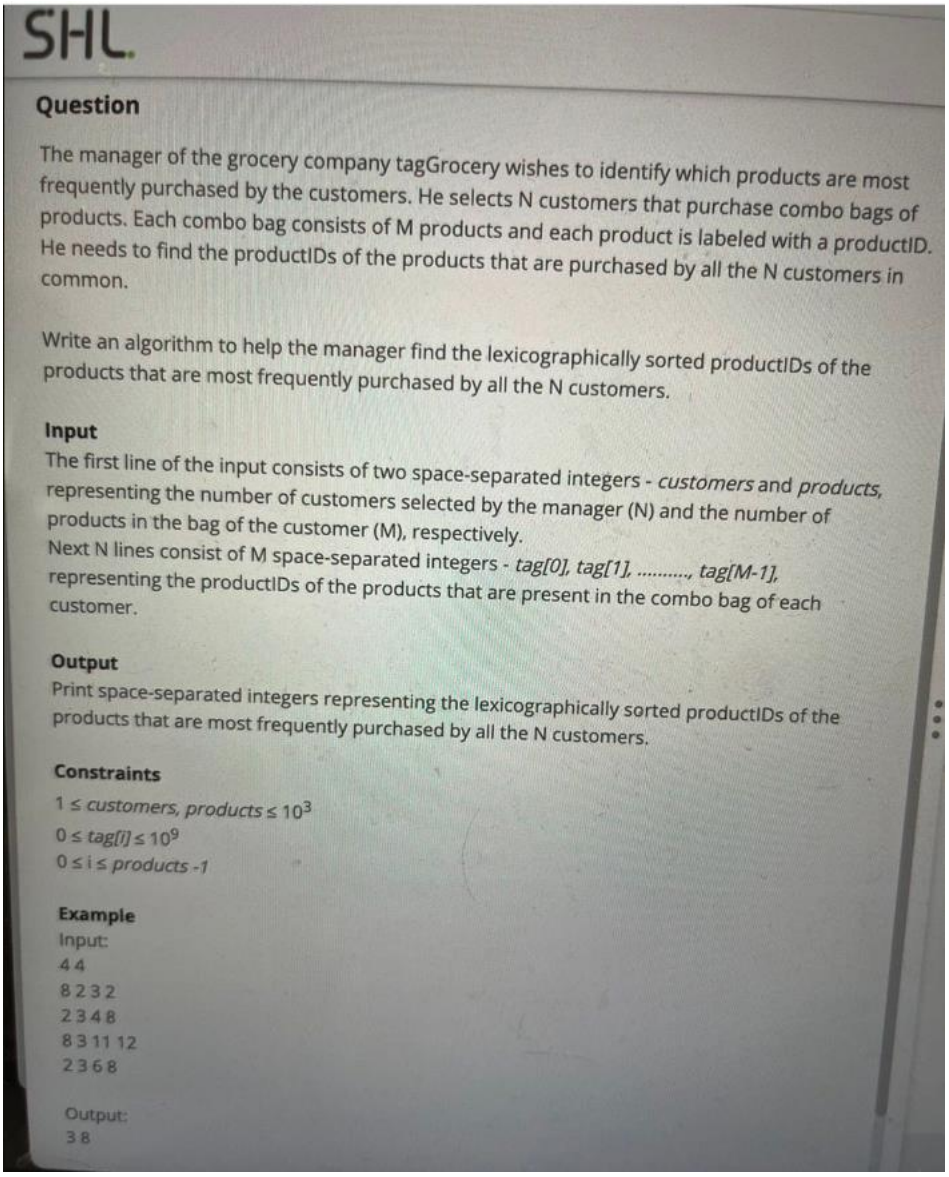
Input:
5
2 4 6 8 10

Output:
2

Explanation:
The largest positive integer that divides all the positive integers 2 4 6 8 10 without a remainder is 2.
So, the output is 2.

Solution: __gcd(m, n)

Problem-2: The manager of the grocery company tagGrocery wishes to identify…



SHL.

Question

The manager of the grocery company tagGrocery wishes to identify which products are most frequently purchased by the customers. He selects N customers that purchase combo bags of products. Each combo bag consists of M products and each product is labeled with a productID. He needs to find the productIDs of the products that are purchased by all the N customers in common.

Write an algorithm to help the manager find the lexicographically sorted productIDs of the products that are most frequently purchased by all the N customers.

Input

The first line of the input consists of two space-separated integers - *customers* and *products*, representing the number of customers selected by the manager (N) and the number of products in the bag of the customer (M), respectively.
Next N lines consist of M space-separated integers - *tag[0], tag[1], ..., tag[M-1]*, representing the productIDs of the products that are present in the combo bag of each customer.

Output

Print space-separated integers representing the lexicographically sorted productIDs of the products that are most frequently purchased by all the N customers.

Constraints

$1 \leq customers, products \leq 10^3$
$0 \leq tag[i] \leq 10^9$
$0 \leq i \leq products - 1$

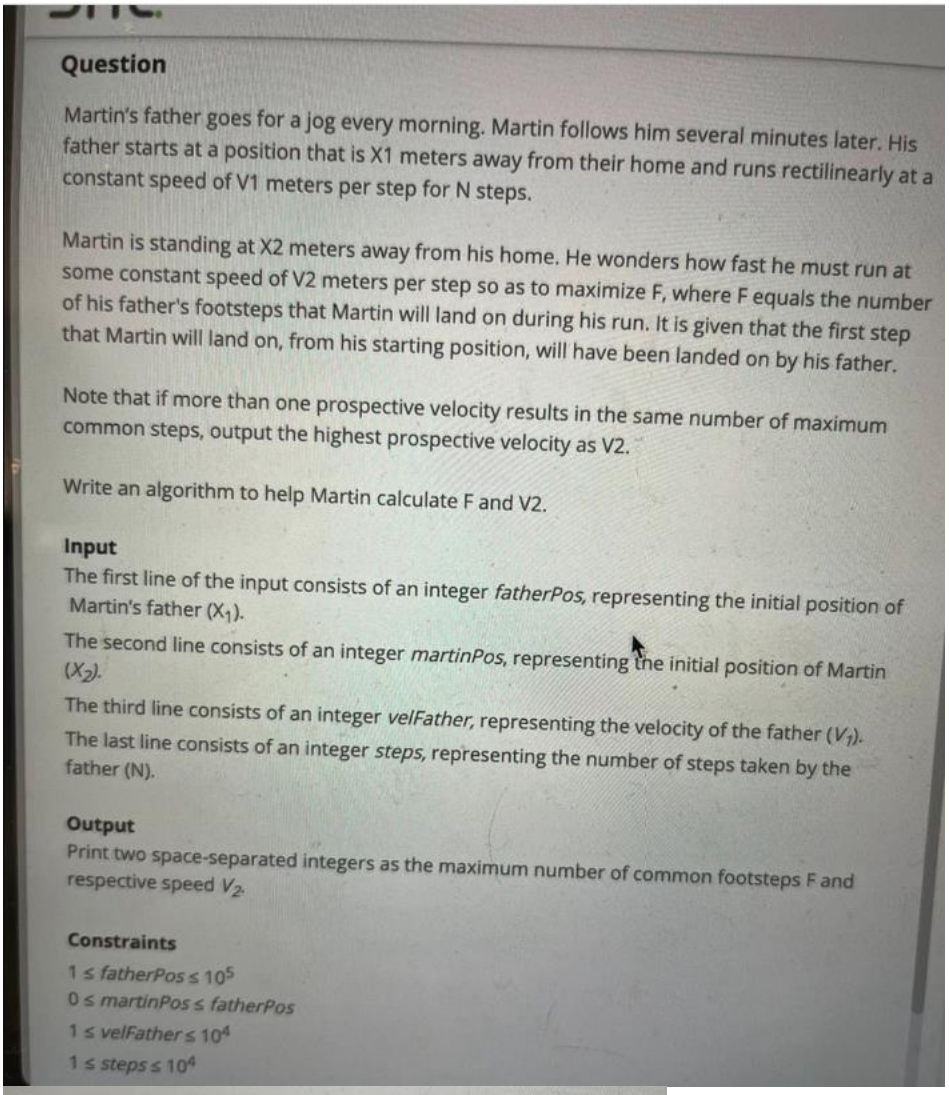
Example

Input:
4 4
8 2 3 2
2 3 4 8
8 3 11 12
2 3 6 8

Output:
3 8

Maintain unordered_map<pid, cnt> and for each customer create new array because customer can purchase same product multiple times thus we don’t want to do cnt++ in unordered_map!

Problem-3: Martin's Father goes for a jog every morning.



Question

Martin's father goes for a jog every morning. Martin follows him several minutes later. His father starts at a position that is X1 meters away from their home and runs rectilinearly at a constant speed of V1 meters per step for N steps.

Martin is standing at X2 meters away from his home. He wonders how fast he must run at some constant speed of V2 meters per step so as to maximize F, where F equals the number of his father's footsteps that Martin will land on during his run. It is given that the first step that Martin will land on, from his starting position, will have been landed on by his father.

Note that if more than one prospective velocity results in the same number of maximum common steps, output the highest prospective velocity as V2.

Write an algorithm to help Martin calculate F and V2.

Input

The first line of the input consists of an integer *fatherPos*, representing the initial position of Martin's father (X_1).

The second line consists of an integer *martinPos*, representing the initial position of Martin (X_2).

The third line consists of an integer *velFather*, representing the velocity of the father (V_1).

The last line consists of an integer *steps*, representing the number of steps taken by the father (N).

Output

Print two space-separated integers as the maximum number of common footsteps F and respective speed V_2.

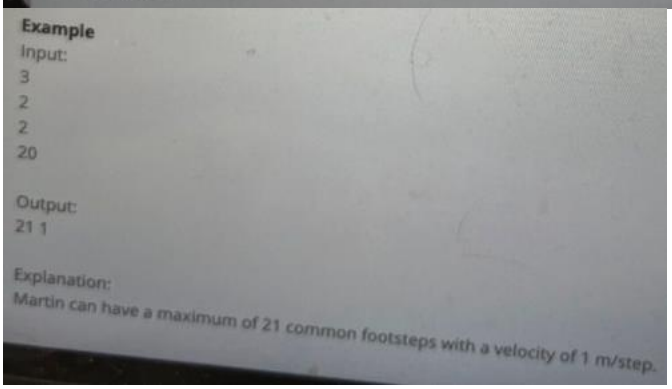
Constraints

$1 \le fatherPos \le 10^5$

$0 \le martinPos \le fatherPos$

$1 \le velFather \le 10^4$

$1 \le steps \le 10^4$



Example

Input:

3
2
2
20

Output:

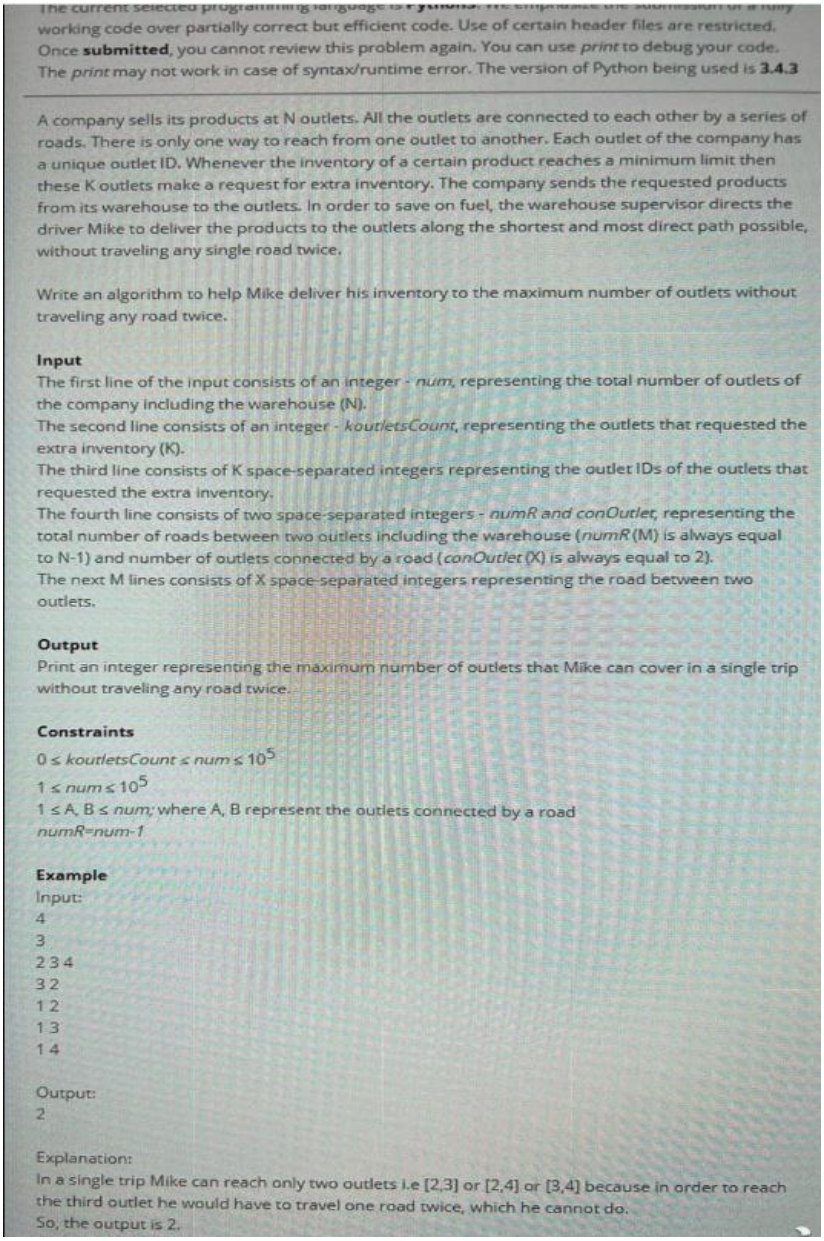
21 1

Explanation:

Martin can have a maximum of 21 common footsteps with a velocity of 1 m/step.

https://leetcode.com/discuss/interview-question/827362/oa-question-2020

Problem-4: A company sells its products at N outlets. All the outlets are connected to each other by a series of roads. There is only one



working code over partially correct but efficient code. Use of certain header files are restricted. Once **submitted**, you cannot review this problem again. You can use *print* to debug your code. The *print* may not work in case of syntax/runtime error. The version of Python being used is **3.4.3**

A company sells its products at N outlets. All the outlets are connected to each other by a series of roads. There is only one way to reach from one outlet to another. Each outlet of the company has a unique outlet ID. Whenever the inventory of a certain product reaches a minimum limit then these K outlets make a request for extra inventory. The company sends the requested products from its warehouse to the outlets. In order to save on fuel, the warehouse supervisor directs the driver Mike to deliver the products to the outlets along the shortest and most direct path possible, without traveling any single road twice.

Write an algorithm to help Mike deliver his inventory to the maximum number of outlets without traveling any road twice.

Input
The first line of the input consists of an integer - *num*, representing the total number of outlets of the company including the warehouse (N).
The second line consists of an integer - *koutletsCount*, representing the outlets that requested the extra inventory (K).
The third line consists of K space-separated integers representing the outlet IDs of the outlets that requested the extra inventory.
The fourth line consists of two space-separated integers - *numR and conOutlet*, representing the total number of roads between two outlets including the warehouse (*numR* (M) is always equal to N-1) and number of outlets connected by a road (*conOutlet* (X) is always equal to 2).
The next M lines consists of X space-separated integers representing the road between two outlets.

Output
Print an integer representing the maximum number of outlets that Mike can cover in a single trip without traveling any road twice.

Constraints
$0 \leq koutletsCount \leq num \leq 10^5$
$1 \leq num \leq 10^5$
$1 \leq A, B \leq num$; where A, B represent the outlets connected by a road
numR=num-1

Example
Input:
4
3
2 3 4
3 2
1 2
1 3
1 4

Output:
2

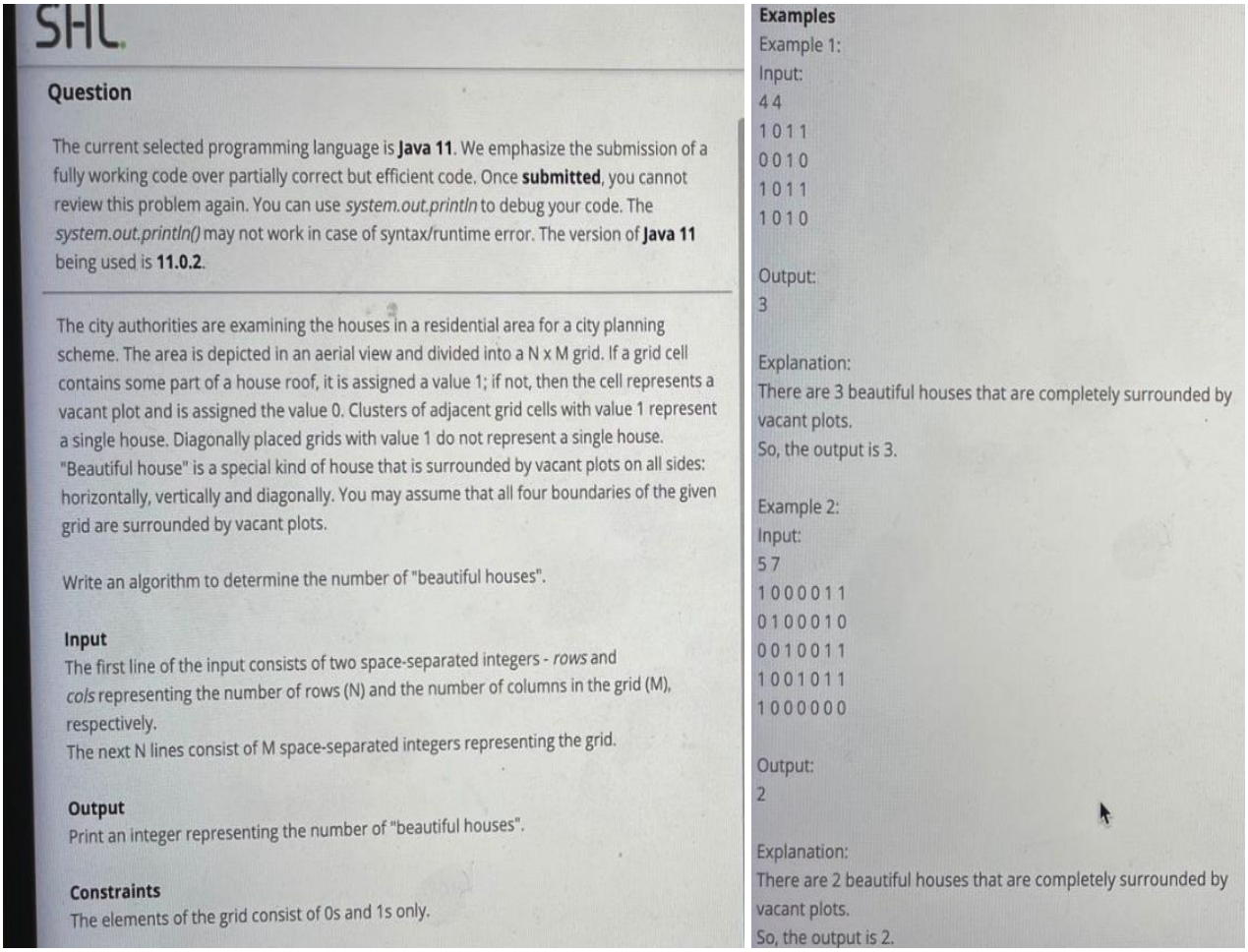
Explanation:
In a single trip Mike can reach only two outlets i.e [2,3] or [2,4] or [3,4] because in order to reach the third outlet he would have to travel one road twice, which he cannot do.
So, the output is 2.

Build the graph (edgelist). And also maintain the degree of vertexes.
Out of given outlets (requested), perform bfs with any of the vertex having degree=1, and pass on the distance to neighbouring nodes, increase the distance only if you encounter the requested node and so on.
At last check the highest distance marked for requested node that will be the answer.

Only 1 BFS required that's it.

```
"""
num is the total number of outlets of the company including the warehouse.
reqOutletsIDs is a list representing the outlet IDs of the outlets that requested the extra inventory.
roadCon is a grid where each row represents the outlets connected by a road.
"""
def maxOutlets(num, reqOutletsIDs, roadCon):
    # Write your code here


    return

def main():
    #input for num
    num = int(input())

    #input for reqOutletsIDs
    reqOutletsIDs = []
    reqOutletsIDs_size = int(input())
    reqOutletsIDs = list(map(int,input().split()))

    #input for roadCon
    roadCon = []
    roadCon_rows,roadCon_cols = map(int, input().split())
    for idx in range(roadCon_rows):
        roadCon.append(list(map(int, input().split())))


    result = maxOutlets(num, reqOutletsIDs, roadCon)
    print(result)

if __name__ == "__main__":
    main()
```

Problem-5: The city authorities are examining the houses in a residential area for a city planning



SHL

Question

The current selected programming language is **Java 11**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *system.out.println* to debug your code. The *system.out.println()* may not work in case of syntax/runtime error. The version of **Java 11** being used is **11.0.2**.

The city authorities are examining the houses in a residential area for a city planning scheme. The area is depicted in an aerial view and divided into a N x M grid. If a grid cell contains some part of a house roof, it is assigned a value 1; if not, then the cell represents a vacant plot and is assigned the value 0. Clusters of adjacent grid cells with value 1 represent a single house. Diagonally placed grids with value 1 do not represent a single house. "Beautiful house" is a special kind of house that is surrounded by vacant plots on all sides: horizontally, vertically and diagonally. You may assume that all four boundaries of the given grid are surrounded by vacant plots.

Write an algorithm to determine the number of "beautiful houses".

Input
The first line of the input consists of two space-separated integers - *rows* and *cols* representing the number of rows (N) and the number of columns in the grid (M), respectively.
The next N lines consist of M space-separated integers representing the grid.

Output
Print an integer representing the number of "beautiful houses".

Constraints
The elements of the grid consist of 0s and 1s only.

Examples
Example 1:
Input:
4 4
1 0 1 1
0 0 1 0
1 0 1 1
1 0 1 0

Output:
3

Explanation:
There are 3 beautiful houses that are completely surrounded by vacant plots.
So, the output is 3.

Example 2:
Input:
5 7
1 0 0 0 0 1 1
0 1 0 0 0 1 0
0 0 1 0 0 1 1
1 0 0 1 0 1 1
1 0 0 0 0 0 0

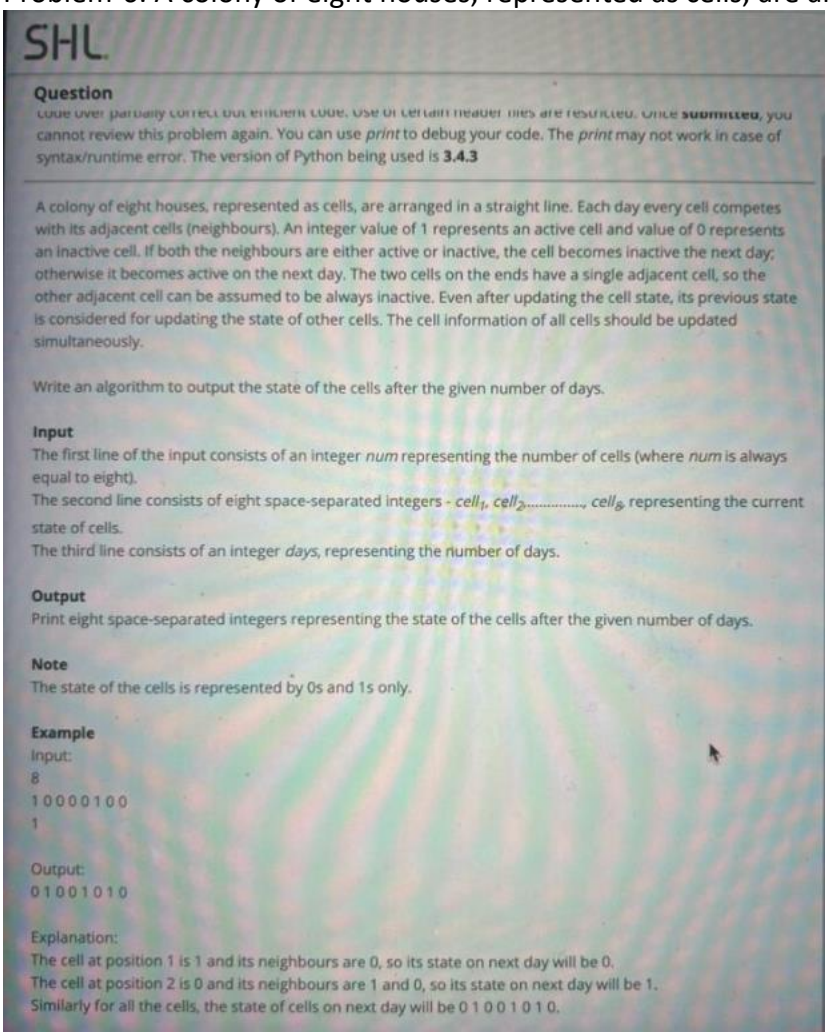
Output:
2

Explanation:
There are 2 beautiful houses that are completely surrounded by vacant plots.
So, the output is 2.

Solution: multi-source BFS, start with any non-zero cell and mark it with that particular number, jitne number bangene vahi hai answer.

Note:Before all that remove all those 1s whos all four directions has 0's! why? Because one 1 is not house, atleast 2 chahiye ones (and vo bhi diagonally nahi, left right up down) mai 1s chahciye.

Problem-6: A colony of eight houses, represented as cells, are arranged in a straight line.



SHL

Question

Code over partially correct but efficient code. Use of certain header files are restricted. Once **submitted**, you cannot review this problem again. You can use *print* to debug your code. The *print* may not work in case of syntax/runtime error. The version of Python being used is **3.4.3**

A colony of eight houses, represented as cells, are arranged in a straight line. Each day every cell competes with its adjacent cells (neighbours). An integer value of 1 represents an active cell and value of 0 represents an inactive cell. If both the neighbours are either active or inactive, the cell becomes inactive the next day; otherwise it becomes active on the next day. The two cells on the ends have a single adjacent cell, so the other adjacent cell can be assumed to be always inactive. Even after updating the cell state, its previous state is considered for updating the state of other cells. The cell information of all cells should be updated simultaneously.

Write an algorithm to output the state of the cells after the given number of days.

Input

The first line of the input consists of an integer *num* representing the number of cells (where *num* is always equal to eight).

The second line consists of eight space-separated integers - $cell_1$, $cell_2$....., $cell_8$, representing the current state of cells.

The third line consists of an integer *days*, representing the number of days.

Output

Print eight space-separated integers representing the state of the cells after the given number of days.

Note

The state of the cells is represented by 0s and 1s only.

Example

Input:

8

1 0 0 0 0 1 0 0

1

Output:

0 1 0 0 1 0 1 0

Explanation:

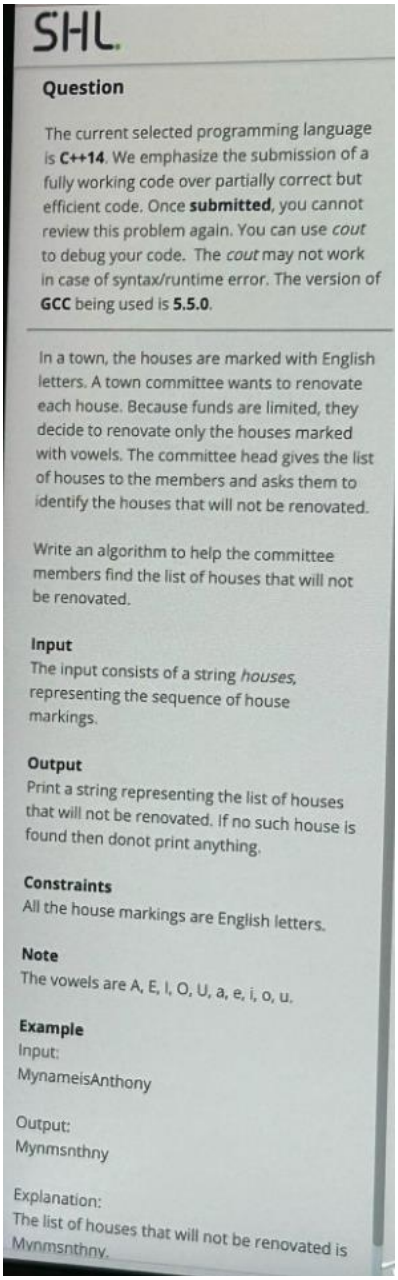
The cell at position 1 is 1 and its neighbours are 0, so its state on next day will be 0.

The cell at position 2 is 0 and its neighbours are 1 and 0, so its state on next day will be 1.

Similarly for all the cells, the state of cells on next day will be 0 1 0 0 1 0 1 0.

Solution: straightway implementation.

Problem-7: In a town, the houses are marked with English letters. A town committee wants to renovate each house.



SHL

Question

The current selected programming language is **C++14**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *cout* to debug your code. The *cout* may not work in case of syntax/runtime error. The version of **GCC** being used is **5.5.0**.

In a town, the houses are marked with English letters. A town committee wants to renovate each house. Because funds are limited, they decide to renovate only the houses marked with vowels. The committee head gives the list of houses to the members and asks them to identify the houses that will not be renovated.

Write an algorithm to help the committee members find the list of houses that will not be renovated.

Input
The input consists of a string *houses*, representing the sequence of house markings.

Output
Print a string representing the list of houses that will not be renovated. If no such house is found then donot print anything.

Constraints
All the house markings are English letters.

Note
The vowels are A, E, I, O, U, a, e, i, o, u.

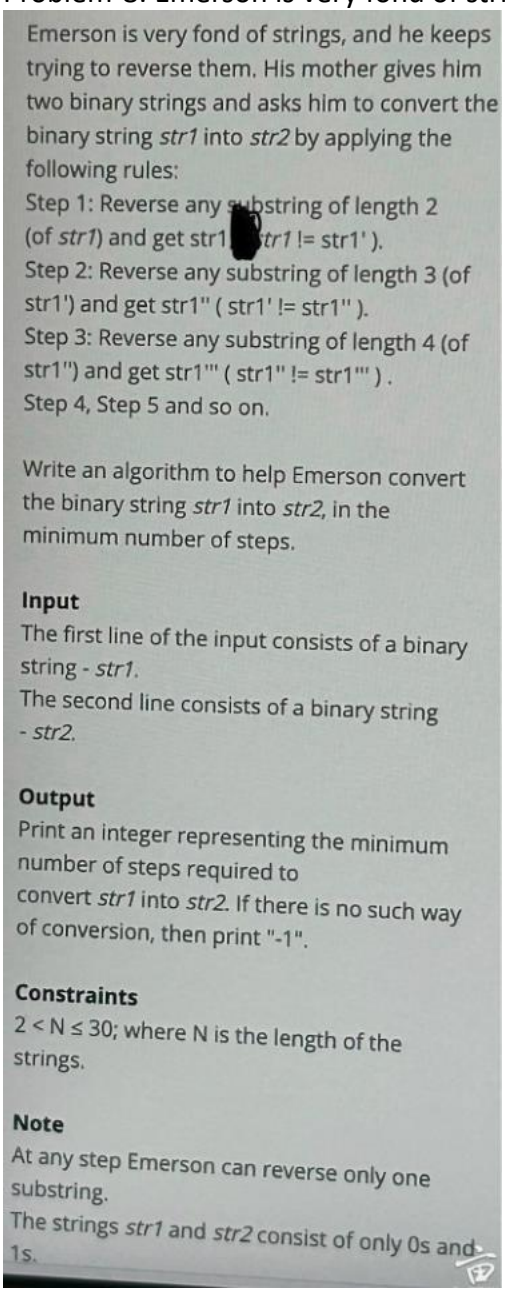
Example
Input:
MynameisAnthony

Output:
Mynmsnthny

Explanation:
The list of houses that will not be renovated is Mynmsnthny.

Solution: straightway implementation.

Problem-8: Emerson is very fond of strings, and he keeps trying to reverse them.



Emerson is very fond of strings, and he keeps trying to reverse them. His mother gives him two binary strings and asks him to convert the binary string *str1* into *str2* by applying the following rules:
Step 1: Reverse any substring of length 2 (of *str1*) and get str1' (str1 != str1').
Step 2: Reverse any substring of length 3 (of str1') and get str1'' (str1' != str1'').
Step 3: Reverse any substring of length 4 (of str1'') and get str1''' (str1'' != str1''') .
Step 4, Step 5 and so on.

Write an algorithm to help Emerson convert the binary string *str1* into *str2*, in the minimum number of steps.

Input
The first line of the input consists of a binary string - *str1*.
The second line consists of a binary string - *str2*.

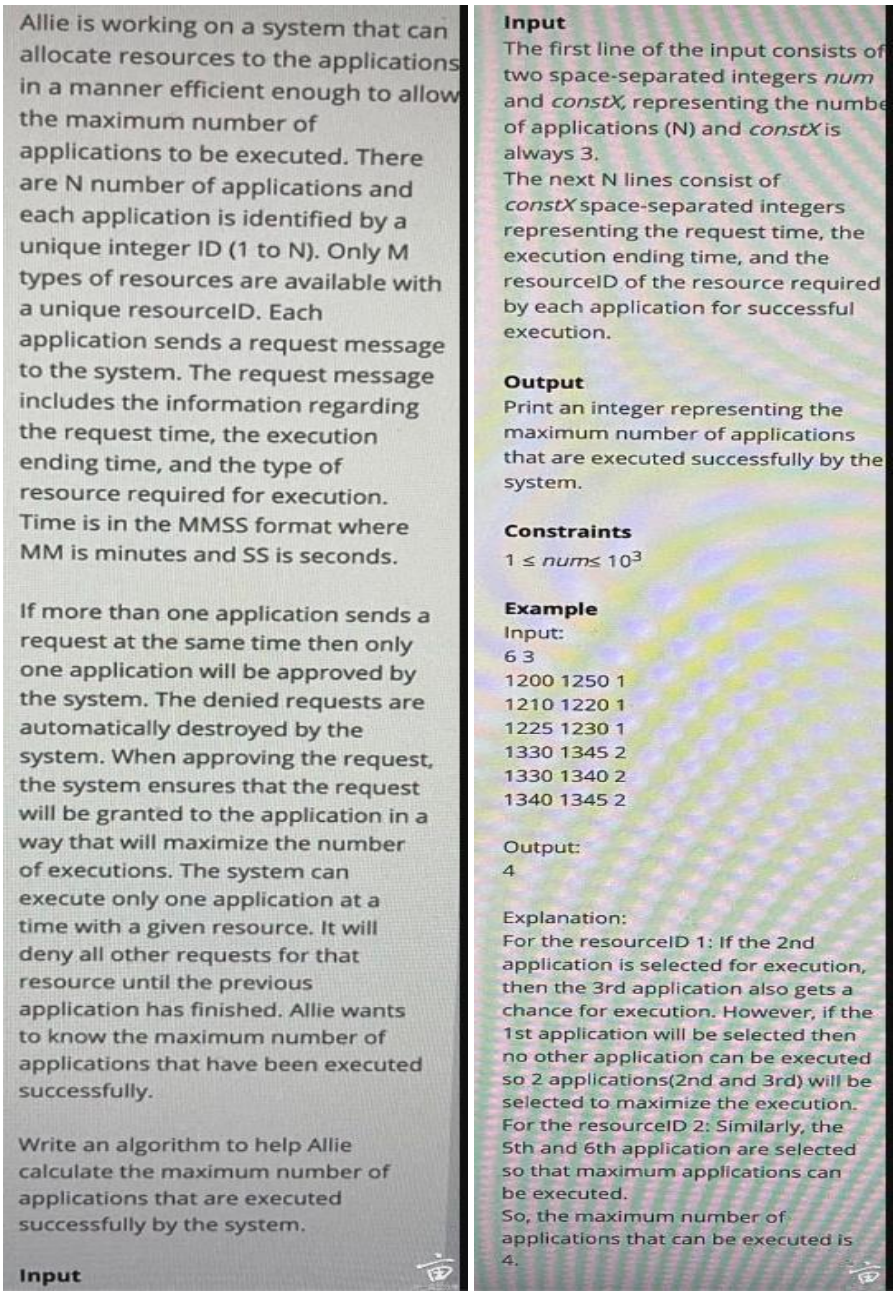
Output
Print an integer representing the minimum number of steps required to convert *str1* into *str2*. If there is no such way of conversion, then print "-1".

Constraints
$2 < N \leq 30$; where N is the length of the strings.

Note
At any step Emerson can reverse only one substring.
The strings *str1* and *str2* consist of only 0s and 1s.

Problem Link: Click here

Problem-9: Allie is working on a system that can allocate resources to the applications in a manner efficient



Allie is working on a system that can allocate resources to the applications in a manner efficient enough to allow the maximum number of applications to be executed. There are N number of applications and each application is identified by a unique integer ID (1 to N). Only M types of resources are available with a unique resourceID. Each application sends a request message to the system. The request message includes the information regarding the request time, the execution ending time, and the type of resource required for execution. Time is in the MMSS format where MM is minutes and SS is seconds.

If more than one application sends a request at the same time then only one application will be approved by the system. The denied requests are automatically destroyed by the system. When approving the request, the system ensures that the request will be granted to the application in a way that will maximize the number of executions. The system can execute only one application at a time with a given resource. It will deny all other requests for that resource until the previous application has finished. Allie wants to know the maximum number of applications that have been executed successfully.

Write an algorithm to help Allie calculate the maximum number of applications that are executed successfully by the system.

Input

Input
The first line of the input consists of two space-separated integers *num* and *constX*, representing the numbe of applications (N) and *constX* is always 3.
The next N lines consist of *constX* space-separated integers representing the request time, the execution ending time, and the resourceID of the resource required by each application for successful execution.

Output
Print an integer representing the maximum number of applications that are executed successfully by the system.

Constraints
$1 \leq num \leq 10^3$

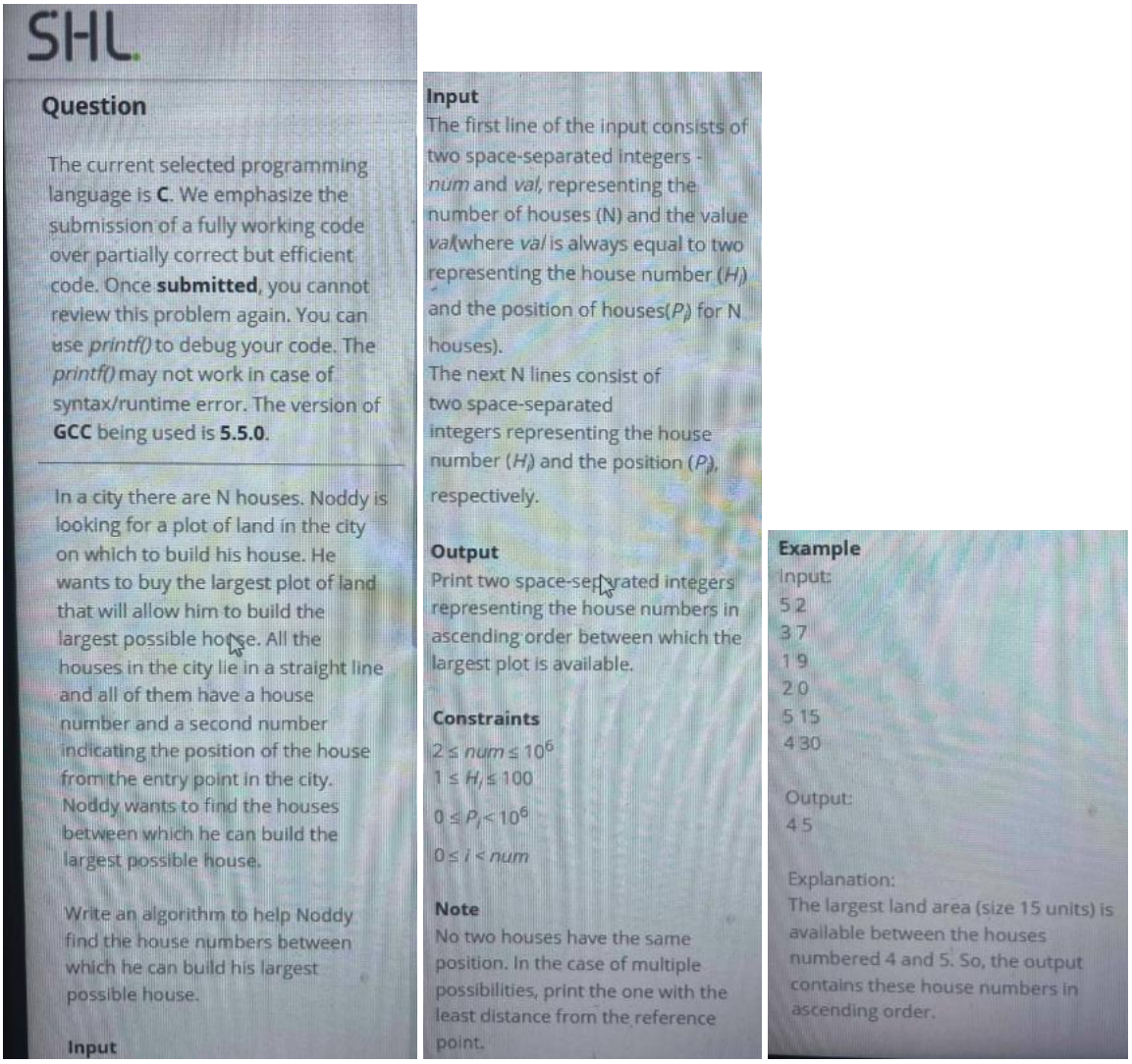
Example
Input:
6 3
1200 1250 1
1210 1220 1
1225 1230 1
1330 1345 2
1330 1340 2
1340 1345 2

Output:
4

Explanation:
For the resourceID 1: If the 2nd application is selected for execution, then the 3rd application also gets a chance for execution. However, if the 1st application will be selected then no other application can be executed so 2 applications(2nd and 3rd) will be selected to maximize the execution.
For the resourceID 2: Similarly, the 5th and 6th application are selected so that maximum applications can be executed.
So, the maximum number of applications that can be executed is 4.

Solution: https://leetcode.com/discuss/interview-question/1621080/SAP-Labs-or-OA-2021-or-Maximum-Executed-Applications/1708623

Problem-10: In a city there are N houses. Noddy is looking for a plot of land in the city on which to build his house.



SHL

Question

The current selected programming language is **C**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *printf()* to debug your code. The *printf()* may not work in case of syntax/runtime error. The version of **GCC** being used is **5.5.0**.

In a city there are N houses. Noddy is looking for a plot of land in the city on which to build his house. He wants to buy the largest plot of land that will allow him to build the largest possible house. All the houses in the city lie in a straight line and all of them have a house number and a second number indicating the position of the house from the entry point in the city. Noddy wants to find the houses between which he can build the largest possible house.

Write an algorithm to help Noddy find the house numbers between which he can build his largest possible house.

Input

Input
The first line of the input consists of two space-separated integers - *num* and *val*, representing the number of houses (N) and the value *val*(where *val* is always equal to two representing the house number (H_i) and the position of houses(P_i) for N houses).
The next N lines consist of two space-separated integers representing the house number (H_i) and the position (P_i), respectively.

Output
Print two space-separated integers representing the house numbers in ascending order between which the largest plot is available.

Constraints
$2 \le num \le 10^6$
$1 \le H_i \le 100$
$0 \le P_i < 10^6$
$0 \le i < num$

Note
No two houses have the same position. In the case of multiple possibilities, print the one with the least distance from the reference point.

Example
Input:
5 2
3 7
1 9
2 0
5 15
4 30

Output:
4 5

Explanation:
The largest land area (size 15 units) is available between the houses numbered 4 and 5. So, the output contains these house numbers in ascending order.

Solution: straight way, vector<pair<int, int>> = {{position, house_no}} sort it by position.

Problem 11: Mary, a physical education teacher, divides her students into different groups and assigns an ID to each group.

Mary, a physical education teacher, divides her students into different groups and assigns an ID to each group. For the group ID, she asks the students to stand in a queue. Each student in the class has a performance factor (PFR). She assigns scores to the students in an unusual way based on their PFR. She gives a score of 5 to a student behind whom is standing at least one student with a higher PFR, behind whom is standing at least one student with a smaller PFR. Next, she gives a score of 10 to a student behind whom is standing a student with a higher PFR, behind whom no student with smaller PFR is standing. Finally, she gives a score of 15 to a student behind whom is standing no student with a higher PFR. The group ID is the sum of scores of the students in the group.

Write an algorithm to find the group ID of a group of students.

Input

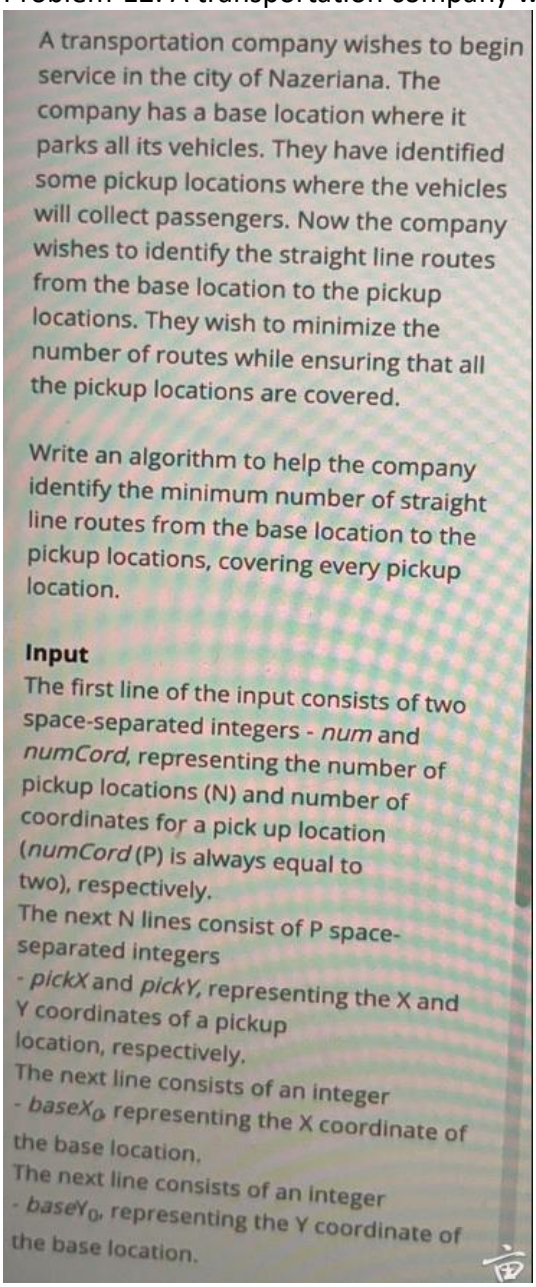
The first line of the input consists of an integer *num*, representing the number of students in a group.
The second line consists of N space-separated integers - *listPFR[0], listPFR[1],.....,listPFR[N-1]* representing the PFR of the students in the order in which they are standing in the queue.

Output

Print an integer representing the group ID of the group of students.

Solution: simple straight implementation, maintain small and larges PFR for each index from right to left and assign score accordingly.

Problem-12: A transportation company wishes to begin service in the city of Nazeriana.



A transportation company wishes to begin service in the city of Nazeriana. The company has a base location where it parks all its vehicles. They have identified some pickup locations where the vehicles will collect passengers. Now the company wishes to identify the straight line routes from the base location to the pickup locations. They wish to minimize the number of routes while ensuring that all the pickup locations are covered.

Write an algorithm to help the company identify the minimum number of straight line routes from the base location to the pickup locations, covering every pickup location.

Input

The first line of the input consists of two space-separated integers - *num* and *numCord*, representing the number of pickup locations (N) and number of coordinates for a pick up location (*numCord* (P) is always equal to two), respectively.

The next N lines consist of P space-separated integers

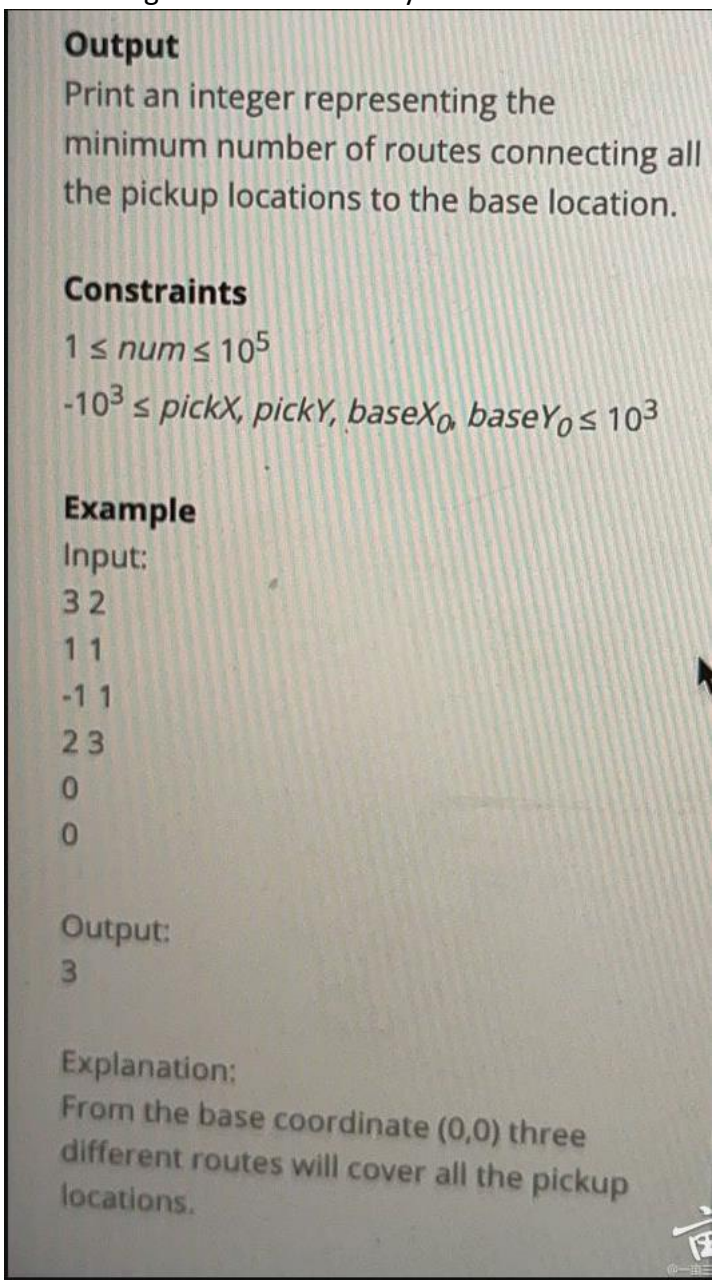
- *pickX* and *pickY*, representing the X and Y coordinates of a pickup location, respectively.

The next line consists of an integer

- $baseX_0$, representing the X coordinate of the base location.

The next line consists of an integer

- $baseY_0$, representing the Y coordinate of the base location.



Output

Print an integer representing the minimum number of routes connecting all the pickup locations to the base location.

Constraints

$1 \le num \le 10^5$

$-10^3 \le pickX, pickY, baseX_0, baseY_0 \le 10^3$

Example

Input:

3 2
1 1
-1 1
2 3
0
0

Output:

3

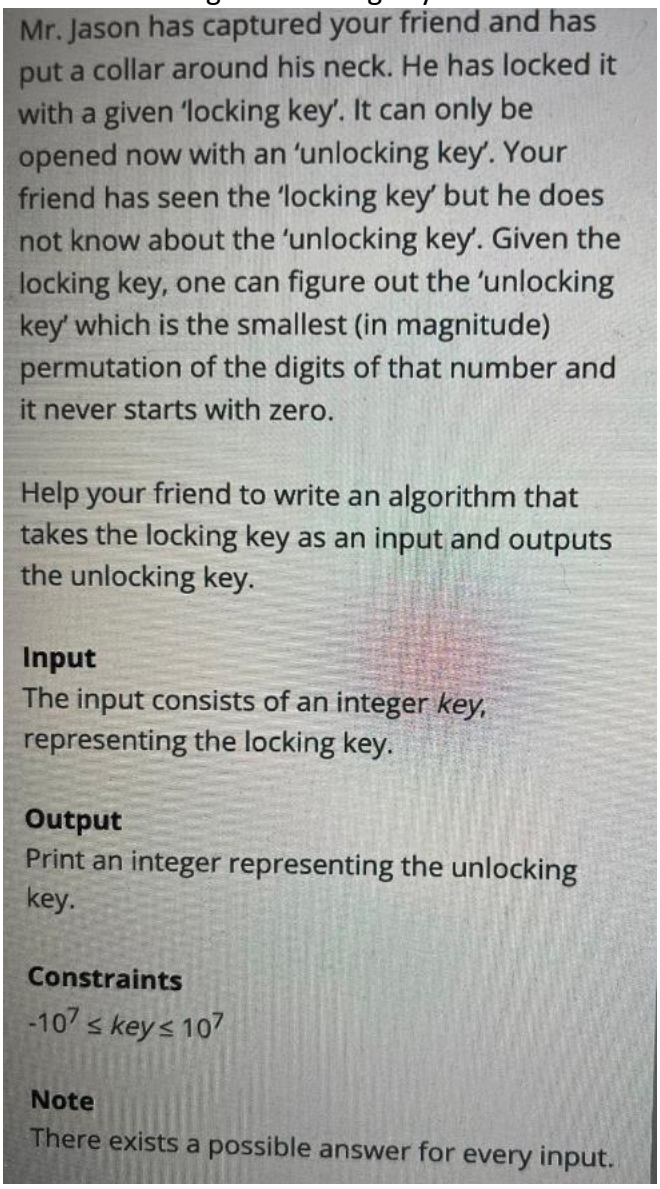
Explanation:

From the base coordinate (0,0) three different routes will cover all the pickup locations.

Good question:

Link1: votrubac1, Learning out of this problem – float division (a/b) (c/d) never compare because of rounding error! Instead follow this way. Link3

Problem-14: Mr. Jason has captured your friend and has put a collar around his neck. He has locked it with a given 'locking key'.



Mr. Jason has captured your friend and has put a collar around his neck. He has locked it with a given 'locking key'. It can only be opened now with an 'unlocking key'. Your friend has seen the 'locking key' but he does not know about the 'unlocking key'. Given the locking key, one can figure out the 'unlocking key' which is the smallest (in magnitude) permutation of the digits of that number and it never starts with zero.

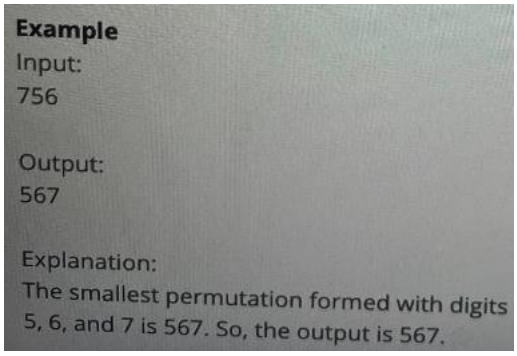
Help your friend to write an algorithm that takes the locking key as an input and outputs the unlocking key.

Input
The input consists of an integer *key*, representing the locking key.

Output
Print an integer representing the unlocking key.

Constraints
$-10^7 \leq key \leq 10^7$

Note
There exists a possible answer for every input.



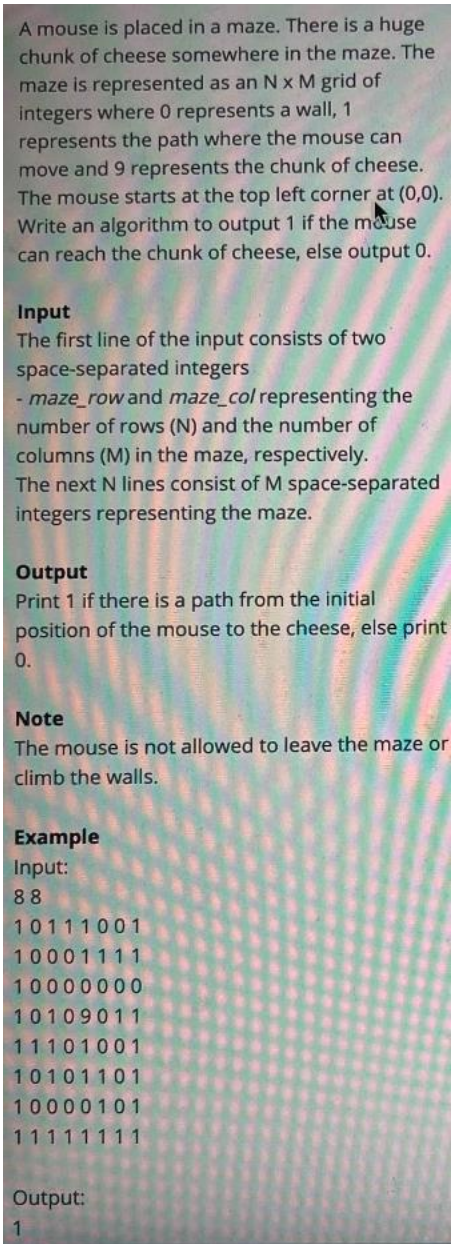
Example
Input:
756

Output:
567

Explanation:
The smallest permutation formed with digits 5, 6, and 7 is 567. So, the output is 567.

Solution: straight way.
Maintain 0-9 indices array mark the occurances, first start with 1 and then 0 and then 2, 3, 4 …9

Problem-15: A mouse is placed in a maze. There is a huge chunk of cheese somewhere in the maze.



A mouse is placed in a maze. There is a huge chunk of cheese somewhere in the maze. The maze is represented as an N x M grid of integers where 0 represents a wall, 1 represents the path where the mouse can move and 9 represents the chunk of cheese. The mouse starts at the top left corner at (0,0). Write an algorithm to output 1 if the mouse can reach the chunk of cheese, else output 0.

Input
The first line of the input consists of two space-separated integers
- *maze_row* and *maze_col* representing the number of rows (N) and the number of columns (M) in the maze, respectively.
The next N lines consist of M space-separated integers representing the maze.

Output
Print 1 if there is a path from the initial position of the mouse to the cheese, else print 0.

Note
The mouse is not allowed to leave the maze or climb the walls.

Example
Input:
8 8
1 0 1 1 1 0 0 1
1 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0
1 0 1 0 9 0 1 1
1 1 1 0 1 0 0 1
1 0 1 0 1 1 0 1
1 0 0 0 0 1 0 1
1 1 1 1 1 1 1 1

Output:
1

Solution: BFS