Question - 1
Missing Prom Dance Partner

During a prom dance, everyone are given a random positive number. People with the same number will have to dance together. However, there is one person who does not have a dance partner. Find the number that person holds.

Example 1:

```
Input: [2,2,1]
Output: 1
```

Example 2:

```
Input: [4,1,2,1,2]
Output: 4
```

Question - 2
Connected Groups

Relationships between people may be represented in a matrix as a series of binary digits. For example, the direct relationships for person *0* with persons *0* through *5* might be shown as *101100*. This means that person *0* knows persons *0, 2* and *3,* the indices of each of the *1* values. A relationship is transitive. In other words, if person *0* knows person *2* and person *2* knows person *3*, then person *0* knows person *3* through person *2*. A *group* is composed of all of the people who know one another, whether directly or transitively.

For example, consider the following relationships matrix:

```
110
110
001
```

Persons *0* and *1* are connected, while person *2* is not. There are *2* groups.

Determine the number of groups represented in a matrix.

Note: The method signatures may vary a little depending on the requirements of your chosen language. For example, C language will

Help

have an argument that represents the number of rows and columns in the matrix.

Function Description

Complete the function *countGroups* in the editor below. The function must return an integer representing the number of groups of people.

countGroups has the following parameter(s):

related[related[0],...related[n-1]]: an array of strings of binary digits *related[i]* that represent direct connections of people

Constraints

- *1 ≤ n ≤ 300*
- *0 ≤ i < n*
- *|related| = n*
- Each *related[i]* contains a binary string of *n* zeros and ones. *related* is a square matrix.

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the square association matrix, *related*.
The next *n* lines each contain a binary string of length *n* that represents a row in the matrix, *related[i]* where *0 ≤ i < n*.

▼ Sample Case 0

Sample Input 0

```
4
1100
1110
0110
0001
```

Sample Output 0

```
2
```

Squares highlighting a connection between two people are highlighted in green. Each of the people is known to self, so they are highlighted in gray.

显示两个人之间的联系的正方形以绿色突出显示。每个人知道自己，因此以灰色突出显示。

Explanation 0

There are *n = 4* people numbered *related[0]* through *related[3]*. There are *2* pairs who directly know each another: *(related[0], related[1])* and *(related[1],related[2])*. Because a relation is transitive, the set of people *{related[0], related[1], related[2]}* is considered a single group. The remaining person, *related[3]*, does not know any other people and is a separate group: *{related[3]}*. There are a total of *2* groups.

两组同学互相之间知道彼此，*{related[0], related[1], related[2]}* 。同学3不知道其他同学，自己一组，*{related[3]}*.

▼ Sample Case 1

Sample Input 1

```
5
10000
01000
00100
00010
00001
```

Sample Output 1

```
5
```

Explanation 1:

No direct relationships are shown so there are *5* groups: *{related[0]}, {related[1]}, {related[2]}, {related[3]}, and {related[4]}*.

5个人互相都不认识彼此，所以共有五组: *{related[0]}, {related[1]}, {related[2]}, {related[3]}, and {related[4]}*.

Question - 3
Shared Interest

Given a graph of friends who have different interests, determine which groups of friends have the most interests in common. Then use a little math to determine a value to return.

The graph will be represented as a series of nodes numbered consecutively from *1* to *friends_nodes*. Friendships have evolved based on interests which will be represented as weights in the graph. Any members who share the same interest are said to be connected by that interest. Once the node pairs with the maximum number of shared interests are determined, multiply the friends_nodes of the resulting node pairs and return the maximal product.

Example

friends_nodes = 4

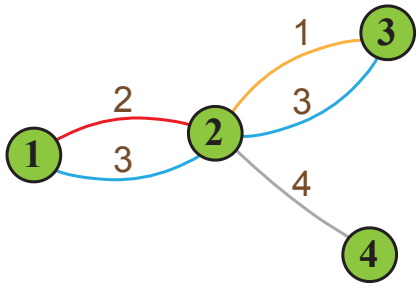
friends_edges = 5

friends_from = [1, 1, 2, 2, 2]

friends_to = [2, 2, 3, 3, 4]

friends_weight = [2, 3, 1, 3, 4]

```
From  To
Weight
1     2
2
1     2
3
2     3
1
2     3
3
2     4
4
```



The graph shows the following connections:

```
                Weight
(Interest)    Connectons
    1            2,3
    2            1,2
    3            1,2,3
    4            2,4
```

- Node pair (2,4) shares only 1 interest (4) and node pair (1,3) shares 1 interest (3).
- Node pair (1,2) shares 2 interests (interests 2 and 3) and node pair (2, 3) shares also 2 interests (interest 1 and 3) . So, the maximum number of shared interests is 2.
- Multiply the friends_nodes of the resulting node pairs : *1 × 2 = 2* and *2 × 3 = 6*.
- The maximal product is 6.

Function Description

Complete the function *maxShared* in the editor below.

maxShared has the following parameter(s):

int *friends_nodes:* number of nodes

int *friends_from[friends_edges]:* the first part of node pairs

int *friends_to[friends_edges]:* the other part of node pairs

int *friends_weight[friends_edges]:* the interests of node pairs

Returns:

int: maximal integer product of all node pairs sharing the most interests.

Constraints

- $2 \le friends\_nodes \le 100$
- $1 \le friends\_edges \le min(200, {}^{(friends\_nodes \times (friends\_nodes - 1))}/_{2})$
- $1 \le friends\_weight[i] \le 100$
- $1 \le friends\_from[i], friends\_to[i] \le friends\_nodes$
- $1 \le friends\_weight[i] \le friends\_edges$
- $friends\_from[i] \ne friends\_to[i]$
- Each pair of friends can be connected by zero or more interests.

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains two space-separated integers *friends_nodes* and *friends_edges*.
Each of the next *friends_edges* lines contains three space-separated integers, *friends_from[i]*, *friends_to[i]* and *friends_weight[i]* where 0 ≤ i < *friends_edges*.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
4 5     →   friends_nodes = 4  friends_edges = 5
1 2 1   →   friends_from = [1,1,2,2,2]  friends_to
= [2,2,3,3,4]  friends_weight = [1,2,1,3,3]
1 2 2
2 3 1
2 3 3
2 4 3
```

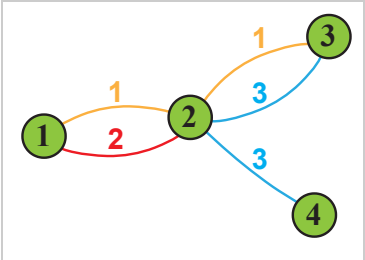
Sample Output 0

```
6
```

Explanation 0

Each pair of *friends_nodes = 4* friends is connected by the following interests:

- Pair *(1, 2)* shares *2* interests (i.e., interests *1* and *2*)
- Pair *(1, 3)* shares *1* interest (i.e., interest *1*)
- Pair *(1, 4)* shares *0* interests
- Pair *(2, 3)* shares *2* interests (i.e., interests *1* and *3*)
- Pair *(2, 4)* shares *1* interest (i.e.,



The pairs connected by the maximal number of interests are *(1, 2)* and *(2, 3)*. Their respective products are *1 × 2 = 2* and *2 × 3 = 6*. The result is the the largest of these values which is *6*.

interest *3*) • Pair *(3, 4)* shares *1* interest (i.e., interest *3*)	

Question - 4
Mirror Image

You write something on a window and you want to know what it look like on another side. On another side, what you wrote would be in a revered order. Write a function that reverse a string that you wrote.

You may assume all the characters consist of printable ascii characters.

Example 1:

```
Input: "hello world"
Output: "dlrow olleh"
```

Example 2:

```
Input: Hannah
Output: hannaH
```

Question - 5
Square Building

A city wants to build a square building on a small island. Engineers draw and gave city an information of the land on the island using 2D binary matrix filled with 0's and 1's, where:

0 represents water.

1 represents land.

Find the largest square land on the island.

Example:

```
Input:

1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0

Output: 4
```

Question - 6
Shortest Path in a Grid with Obstacles Elimination

Given a `m * n` grid, where each cell is either `0` (empty) or `1` (obstacle). In one step, you can move up, down, left or right from and to an empty cell.
Return the minimum number of steps to walk from the upper left corner `(0, 0)` to the lower right corner `(m-1, n-1)` given that you can eliminate **at most** `k` obstacles. If it is not possible to find such walk return -1.

Example 1:

```
Input:
grid =
[[0,0,0],
 [1,1,0],
 [0,0,0],
 [0,1,1],
 [0,0,0]],
k = 1
Output: 6
Explanation:
The shortest path without eliminating any
obstacle is 10.
The shortest path with one obstacle elimination
at position (3,2) is 6. Such path is (0,0) ->
(0,1) -> (0,2) -> (1,2) -> (2,2) -> (3,2) ->
(4,2).
```

Example 2:

```
Input:
grid =
[[0,1,1],
 [1,1,1],
 [1,0,0]],
k = 1
Output: -1
Explanation:
We need to eliminate at least two obstacles to
find such a walk.
```

Constraints:

- `grid.length == m`
- `grid[0].length == n`
- `1 <= m, n <= 40`
- `1 <= k <= m*n`
- `grid[i][j] == 0 or 1`
- `grid[0][0] == grid[m-1][n-1] == 0`

Question - 7

Buying/Selling Airline tickets

Imagine you are an agent who sells airline tickets. You make money by buying cheap tickets and hope to sell at a higher price to customers. Lucky for you, Adam who is your best friend works on Wall Street and happened to know the price of tickets for the next fifteen days. Design an algorithm so that you find the maximum amount of profit.

Note: Be wary that you cannot sell a ticket before buying one.

Example 1:

```
Input: [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on
day 5 (price = 6), profit = 6-1 = 5.
             Not 7-1 = 6, as selling price needs
to be larger than buying price.
```

Example 2:

```
Input: [7,6,4,3,1]
Output: 0
Explanation: No transaction is done.
```

Question - 8
Delivery Robot

An online delivery company developed a new AI delivery robot and want to test on the street. You are a software engineer in this AI development team. Your task is to determine how many unique paths would it be for robot to reach to the destination. The team will use your result to verify the efficiency of the robot.

A robot always starts at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).
The robot can only move either down or right at any point in time.
The destination is always at the bottom-right corner of the grid (marked 'Finish' in the diagram below).
There are some obstacles on the grids. An obstacle and empty space is marked as `1` and `0` respectively in the grid (see the reference on the example)

Note: *m* and *n* will be at most 100

Example 1:

```
Input:
[
  [0,0,0],
  [0,1,0],
  [0,0,0]
]
Output: 2
Explanation:
There is one obstacle in the middle of the 3x3
grid above.
```

```
There are two ways to reach the bottom-right
corner:
1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right
```