1. There is a ball in a maze with empty spaces and walls. The ball can go through empty spaces by rolling up,down,left or right, but it won't stop rolling until hitting a wall. When the ball stops, it could choose the next direction.
Given the ball's start position, the destination and the maze, find the shortest distance for the ball to stop at the destination. The distance is defined by the number of empty spacestraveled by the ball from the start position (excluded) to the destination (included). If the ball cannot stop at the destination, return -1.
The maze is represented by a binary 2D array. 1 means the wall and 0 means the empty space. You may assume that the borders of the maze are all walls. The start and destination coordinates are represented by row and column indexes.

Example 1:
Input 1: a maze represented by a 2D array
0 0 1 0 0
0 0 0 0 0
0 0 0 1 0
1 1 0 1 1
0 0 0 0 0

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (4, 4)

Output: 12
Explanation: One shortest way is : left -> down -> left -> down -> right -> down -> right.
The total distance is 1 + 1 + 3 + 1 + 2 + 2 + 2 = 12.

Example 2:
Input 1: a maze represented by a 2D array

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

Input 2: start coordinate (rowStart, colStart) = (0, 4)
Input 3: destination coordinate (rowDest, colDest) = (3, 2)
Output: -1

Explanation: There is no way for the ball to stop at the destination.
Note:
There is only one ball and one destination in the maze.
Both the ball and the destination exist on an empty space, and they will not be at the same position initially.
The given maze does not contain border (like the red rectangle in the example pictures), but you could assume the border of the maze are all walls.
The maze contains at least 2 empty spaces, and both the width and height of the maze won't exceed 100.
==========
2. Given an array of n elements, write an algorithm to rotate it right by k element without using any other array. In other words rotate an array in place.

Given an array, rotate the array to the right by k steps, where k is non-negative.
Example 1:
Input: [1,2,3,4,5,6,7] and k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]

Example 2:
Input: [-1,-100,3,99] and k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]
==========

3. Given a non-empty string, encode the string such that its encoded length is the shortest.
The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times.
Note:
k will be a positive integer and encoded string will not be empty or have extra space.
You may assume that the input string contains only lowercase English letters. The string's length is at most 160.
If an encoding process does not make the string shorter, then do not encode it. If there are several solutions, return any of them is fine.

Example 1:
Input: "aaa"
Output: "aaa"
Explanation: There is no way to encode it such that it(3[a]) is not shorter than the input string, so we do not encode it.

Example 2:
Input: "aaaaa"
Output: "5[a]"
Explanation: "5[a]" is shorter than "aaaaa" by 1 character.

Example 3:
Input: "aaaaaaaaaa"
Output: "10[a]"
Explanation: "a9[a]" or "9[a]a" are also valid solutions, both of them have the same length = 5, which is the same as "10[a]".

Example 4:
Input: "aabcaabcd"
Output: "2[aabc]d"
Explanation: "aabc" occurs twice, so one answer can be "2[aabc]d".

Example 5:
Input: "abbbabbbcabbbabbbc"
Output: "2[2[abbb]c]"
Explanation: "abbbabbbc" occurs twice, but "abbbabbbc" can also be encoded to "2[abbb]c", so one answer can be "2[2[abbb]c]".
==========
4. You have three stacks of cylinders where each cylinder has the same diameter, but they may vary in height. You can change the height of a stack by removing and discarding its topmost cylinder any number of times.
Find the maximum possible height of the stacks such that all of the stacks are exactly the same height. This means you must remove zero or more cylinders from the top of zero or more of the three stacks until they're all the same height, then print the height. The removals must be performed in such a way as to maximize the height.
Note: An empty stack is still a stack.
Input Format
The first line contains three space-separated integers, n1, n2, and n3, describing the respective number of cylinders in stacks 1, 2, and 3. The subsequent lines describe the respective heights of each cylinder in a stack from top to bottom:
The second line contains n1 space-separated integers describing the cylinder heights in stack 1.
The third line contains n2 space-separated integers describing the cylinder heights in stack 2.
The fourth line contains n3 space-separated integers describing the cylinder heights in stack 3.

Output Format
Print a single integer denoting the maximum height at which all stacks will be of equal height.
Sample Input
5 3 4
3 2 1 1 1
4 3 2
1 1 4 1
Sample Output
5
==========
5. Given two arrays that represent the arrival and departure times of trains, the task is to find the minimum number of platforms required so that no train waits.
Example-1:
Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

Output: 3
Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Example-2:
Input: arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}
Output: 1
Explanation: Only one platform is needed.
==========