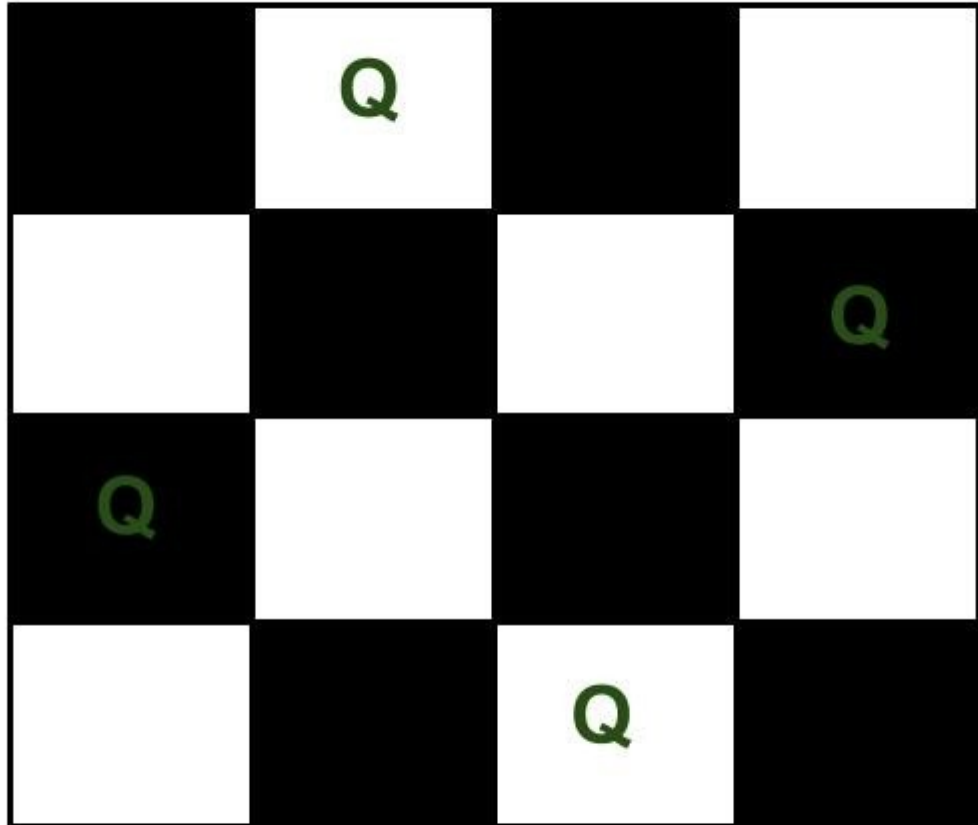


## N-Queen Ways

You are given an empty chess board of size  $N \times N$ . Find the number of ways to place  $N$  queens on the board, such that no two queens can kill each other in one move. A queen can move vertically, horizontally and diagonally.



### **Input Format:**

In the function you are given an integer  $N$  as a parameter.

### **Output Format:**

Return the number of possible solutions.

**Constraints:** $1 \leq N \leq 10$ **Sample Testcase 1:****Input:**

4

**Output:**

2

**Sample Testcase 2:****Input:**

1

**Output:**

1

**Solution:** nQueenProblem.cpp**Rat and Mice**

You are given an  $N \times M$  grid. Each cell  $(i,j)$  in the grid is either blocked, or empty. The rat can move from a position towards left, right, up or down on the grid.

Initially rat is on the position  $(1,1)$ . It wants to reach position  $(N,M)$  where its mice is waiting for. There exists a unique path in the grid. Find that path and help the rat reach its mice.

**Input Format**

Given vector of strings representing a grid with  $N$  rows and  $M$  columns.

'X' in position  $(i,j)$  denotes that the cell is blocked and 'O' denotes that the cell is empty.

**Output Format**

Print  $N$  lines, containing  $M$  integers each. A 1 at a position  $(i,j)$  denotes that the  $(i,j)$ th cell is covered in the path and a 0 denotes that the cell is not covered in the path.

**Sample Input**

5 4OX00000XXOXOX00XXX00

**Sample Output**

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

**Solution:** ratMaze.cpp

## **Word Break Problem**

Given a str which is a valid sentence without any spaces between the words and a dictionary of valid English words, find all possible number of ways to break the sentence in individual dictionary words.

### **Input Format**

In the function a string str and a vector of strings representing dictionary is passed

### **Output Format**

Return an integer representing the total no. of ways

### **Sample Input**

dictionary = { i, like, sam, sung, samsung, mobile}str = "ilikesamsungmobile"

### **Sample Output**

2

### **Explanation**

Following are the 2 possible ways to break the string

i like sam sung mobile i like samsung mobile

**Solution:** wordBreak.cpp

## **Unique Subset**

Given an integer array nums that may contain duplicates, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **ascending order**.

### **Input Format**

In the function an integer vector is passed.

### **Output Format**

Return a vector of vectors containing unique subsets.

### **Sample Input**

[1, 2, 2]

### Sample Output

```
[[ ], [1], [1,2], [1,2,2], [2], [2,2]]
```

**Solution:** uniqueSubsets.cpp

## Word Search

Given an  $m \times n$  grid of characters `board` and a string `word`, return `true` if `word` exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

### Input Format

In the function 2-d vector of character is passed.

### Output Format

Return **true** if the word is present otherwise **false**.

### Sample Input

```
board = [ ["A","B","C","E"], ["S","F","C","S"], ["A","D","E","E"] ] word = "ABCCED"
```

### Sample Output

```
true
```

### Explanation

A	B	C	E
S	F	C	S
A	D	E	E

**Solution:** wordSearch.cpp

## Modulo Exponentiation

You are given two numbers a and b, your task is to compute a raised to power b.

As this value can be very large compute it to modulo 1000000007.

### **HINT : Use Modulo Properties**

$(m * n) \% p$  has a very interesting property:  $(m * n) \% p = ((m \% p) * (n \% p)) \% p$

### **Expected Time Complexity:**

$O(\log b)$

### **Input Format:**

In the function, 2 integers a and b are passed.

### **Output Format:**

Return a single integer which is equivalent to  $(a^b)$  modulo 1000000007.

### **Constraints:**

$1 \leq a, b \leq 50$

### **Sample Testcase:**

#### **Input:**

5 3

#### **Output:**

125

**Solution:** moduloExp.cpp

## Sudoku Solver

You are given a 9X9 2D array which is partially filled. Your aim is to completely fill the grid from digits from 1 to 9 such that every row, column and sub-grid of size 3X3 (as shown) contains all digits from 1 to 9 exactly once.

### **Input**

#### **Format:**

Input will have  
2D vector(9X9  
matrix)

	3					9		
		6						
			2	4	1		3	
			9			7		
					2			4
	8			7			2	
8	5							
	9		7		4			
					6			1

1	3	2	5	6	7	9	4	8
5	4	6	3	8	9	2	1	7
9	7	8	2	4	1	6	3	5
2	6	4	9	1	8	7	5	3
7	1	5	6	3	2	8	9	4
3	8	9	4	7	5	1	2	6
8	5	7	1	2	3	4	6	9
6	9	1	7	5	4	3	8	2
4	2	3	8	9	6	5	7	1

containing integers from 0 to 9 where 0 represents empty cell and other integers represent already filled cells.

**Output Format:**

Return a solved 2D vector(matrix).

**Sample Testcase:**

**Input:**

030000900

006000000

000241030

000900700

000002004

080070020

850000000

090704000

000006001

**Output:**

132567948

546389217

978241635

264918753

715632894

389475126

857123469

691754382

423896571

**Solution:** sudokuSolver.cpp

## **Longest Possible Route**

Given an M x N matrix, with a few hurdles arbitrarily placed, calculate the length of longest possible route possible from top left position (0,0) to bottom right position (m,n) within the matrix.

We are allowed to move to only adjacent cells which are not hurdles. The route cannot contain any diagonal moves and a location once visited in a particular path cannot be visited again.

**Input Format:**

In the function you are given number of rows M, number of columns N and 2D vector (M X N) consisting of 0s and 1s where 0s represent the hurdles.

**Output Format:**

A single integer representing the length of longest path.

**Constraints:**

$1 \leq M, N \leq 10$

**Sample Testcase:**

**Input:**

3 3

1 1 1

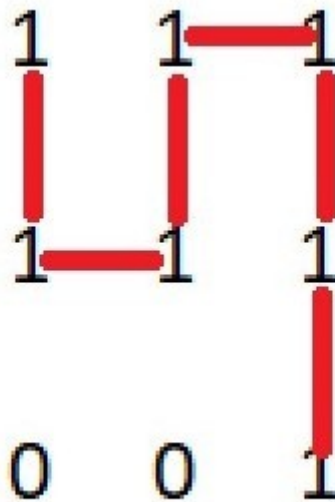
1 1 1

0 0 1

**Output:**

6

**Explanation:**



Here the longest possible path has a path length of 6.

**Solution:** longestRoute.cpp