**`Assignments, W1D5**

Algorithmic Coding:

Problem 1: <https://leetcode.com/problems/n-queens-ii/>- Recursion, Backtracking

Problem 2: Recursion, Backtracking

Given a robot cleaner in a room modeled as a grid.

Each cell in the grid can be empty or blocked.

The robot cleaner with 4 given APIs can move forward, turn left or turn right. Each turn it made is 90 degrees.

When it tries to move into a blocked cell, its bumper sensor detects the obstacle and it stays on the current cell.

Design an algorithm to clean the entire room using only the 4 given APIs shown below.

interface Robot {

// returns true if next cell is open and robot moves into the cell.

// returns false if next cell is obstacle and robot stays on the current cell.

boolean move();

// Robot will stay on the same cell after calling turnLeft/turnRight.

// Each turn will be 90 degrees.

void turnLeft();

void turnRight();

// Clean the current cell.

void clean();

}

Example:

Input:

room = [

[1,1,1,1,1,0,1,1],

[1,1,1,1,1,0,1,1],

[1,0,1,1,1,1,1,1],

[0,0,0,1,0,0,0,0],

[1,1,1,1,1,1,1,1]

],

row = 1,

col = 3

Explanation:

All grids in the room are marked by either 0 or 1.

0 means the cell is blocked, while 1 means the cell is accessible.

The robot initially starts at the position of row=1, col=3.

From the top left corner, its position is one row below and three columns right.

Notes:

1. The input is only given to initialize the room and the robot's position internally. You must solve this problem "blindfolded". In other words, you must control the robot using only the mentioned 4 APIs, without knowing the room layout and the initial robot's position.
2. The robot's initial position will always be in an accessible cell.
3. The initial direction of the robot will be facing up.
4. All accessible cells are connected, which means the all cells marked as 1 will be accessible by the robot.
5. Assume all four edges of the grid are all surrounded by wall.

----------

Given an array of integers nums (all unique), sort the array in ascending order

Problem 3: Insertion Sort

Problem 4: Merge Sort

Problem 5: Quick Sort

Object Oriented Programming:

Problem 1: Multiple Modules

Store the User class in one module, and store the Privileges and Admin classes in a separate module. In a separate file, create an Admin instance and call show\_privileges() to show that everything is still working correctly.

Problem 2: Dice

The module random contains functions that generate random numbers in a variety of ways. The function randint() returns an integer in the range you provide. the following code returns a number between 1 and 6:

from random import randint

x = randint(1, 6)

Make a class Die with one attribute called sides, which has a default value of 6. Write a method called roll\_die() that prints a random number between 1 and the number of sides the die has. Make a 6-sided die and roll it 10 times.

Make a 10-sided die and a 20-sided die. Roll each die 10 times.