Maze

Initial:

- You begin at the starting position (‘S’)
- You position in (x, y) coordinates is in structure pos.x, pos,y
- Call solveMaze(pos);

On each recursion:

- If maze grid at current position is ‘G’ then you are done and return true
- If any of these are true, return false:
    - If maze grid at current position is a Wall, this is an invalid
    - If you have already visited maze grid at current position (grid value is ‘.’), this is an invalid position
    - If maze position is outside of maze, this is an invalid position
- Drop a breadcrumbs (‘.’) to indicate that you have visited this position
- Display maze grid
- Define and initialize the four points of the compass relative to current position
    - `N.x = pos.x; N.y = pos.y-1;`
    - `S.x = pos.x; S.y = pos.y+1;`
    - `E.x = pos.x-1; E.y = pos.y;`
    - `W.x = pos.x+1; W.y = pos.y;`
- Call recursion for each of N, S, E, W
    - If return is true
        - Drop solution path indicator ‘*’ This is the “backtrace” path that is the implicit solution provided by the recursion process.
        - Display maze grid
        - Return true
    - If return is false
        - Erase breadcrumb
        - Return false

Tower of Hanoi

The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883. He was inspired by a legend that tells of a Hindu temple where the puzzle was presented to young priests. At the beginning of time, the priests were given three poles and a stack of 64 gold disks, each disk a little smaller than the one beneath it. Their assignment was to transfer all 64 disks from one of the three poles to another, with two important constraints, indicated in Rules, below.

The priests worked very efficiently, day and night, moving one disk every second. When they finished their work, the legend said, the temple would crumble into dust and the world would vanish.

Initial

- All disks are correctly ordered and on pole number 1 (index 0 in C).

Rules:

- Move one disk at a time, and
- Never place a larger disk on top of a smaller one.

Goal

- All disks are correctly ordered and on pole number 3 (index 2 in C).
- Bonus: should take (2^N)-1 moves.

Hints

- Tower data structure is
    - char grid[NUM_COLUMNS][HEIGHT];
    - where first index is pole number (0, 1, 2) and second index is number of disks
- “number of disks” is number of disks in problem.
    - If it is decremented in recursion process we are effectively solving a smaller problem.
    - When “number of disks” is zero, just return from recursion
- Consider the following solution for one, two and three disks, illustrated below.
- Label positions: 0, 1, 2, or left, middle, right
- Recursion function call is moveTower(num_disks, left, middle, right)
- Recursion function action with 0 disks just return
- Recursion action with 1 disks is as shown below

```
moveTower(1, left, middle, right)
        moveTower(0, …) //args don’t matter since we just return
        move top disk rom left to right
        moveTower(0, …) //args don’t matter since we just return
```

- Recursion action with 2 disks is as shown below. Think about why the order of arguments is changed for the two recursive calls inside the moveTower()

  moveTower(2, left, middle, right)
  moveTower(1, left, right, middle)
  move top disk from left to right
  moveTower(1, middle, left, right)

```
One disk
1
+---+---+
            Move number
1------->   1
Final
         1
+---+---+


Two disks
1
2
+---+---+
            Move number
1--->       1
2------->   2
    1--->   3
Final
         1
         2
+---+---+


Three disks
1
2
3
+---+---+
            Move number
1------->   1
2--->       2
    <---1   3
3------->   4
<---1       5
    2--->   6
1------->   7

Final
         1
         2
         3
+---+---+
```