Ryan Mysliwiec

1a)
After looking at the entire grid, the algorithm splits it up into 4 quadrants. Then it looks for the one with the filled in square. An L-shaped piece is then placed at the center of the overall grid, covering the quadrants that don't yet have a filled piece. The algorithm is then recursively implemented until the entire board is filled.

1b)
Proof: After one square is filled, the rest can filled with 3-block carpet pieces in the given 4 combinations. The board is of size $2^n x 2^n$

Base Case: When n = 1, the board is of size 2x2. When the specified square is filled, the other carpet piece will fill the rest of the board.

Inductive Hypothesis: Assume true for k < n such that $2^k x 2^k$, with one square filled in. This board can be filled by $(2^k-1)/3$ L-shaped carpet pieces.

Inductive Step: A grid with the size of $2^{k+1} x 2^{k+1}$ is made up of 4 $2^k x 2^k$ quadrants. The filled square is in one of those quadrants. When an L-shaped piece is placed at the center of the $2^{k+1} x 2^{k+1}$ board, each quadrant not containing the currently filled square are now "missing" a square as well, where each size is $2^k x 2^k$. Thus, when applying the Inductive Hypothesis, the rest of the board is able to be filled with the previously mentioned L-shaped pieces.

1c)
$f(n) = 1 = n^0 \quad n^{\log_2 4} = n^{1/2}$

$a/b^c = 4/2^0 = 4 > 1 \quad \Rightarrow \quad$ Case 1

$\varepsilon = ½$

$n^0 = n^{1/2 - ½}$

$T(n) = \theta(n^{1/2})$

1d)

```
import sys

def fill_carpet(y_topLeft, x_topLeft, size, y, x):
        if size == 2:
                if x_topLeft == x:
                        if y_topLeft == y:
                                print ('{} {} {}'.format(y_topLeft + 1, x_topLeft + 1, 4))
                        else:
                                print '{} {} {}'.format(y_topLeft, x_topLeft + 1, 3)
                else:
                        if y_topLeft == y:
                                print '{} {} {}'.format(y_topLeft + 1, x_topLeft, 1)
```

Ryan Mysliwiec

```
                else:
                        print '{} {} {}'.format(y_topLeft, x_topLeft, 2)
        else:
                x_center = x_topLeft + (size/2)
                y_center = y_topLeft + (size/2)


                if x < x_center:
                        x_lowR = x_upR = x_center
                        y_lowR = y_center
                        y_upR = y_center - 1
                        if y < y_center:
                                x_lowL = x_center - 1
                                y_lowL = y_center
                                x_upL = x
                                y_upL = y
                                print '{} {} {}'.format(y_center, x_center, 4)
                        else:
                                x_lowL = x
                                y_lowL = y
                                x_upL = x_center - 1
                                y_upL = y_center - 1
                                print '{} {} {}'.format(y_center - 1, x_center, 3)
                else:
                        x_lowL = x_upL = x_center - 1
                        y_lowL = y_center
                        y_upL = y_center - 1
                        if y < y_center:
                                x_lowR = x_center
                                y_lowR = y_center
                                x_upR = x
                                y_upR = y
                                print '{} {} {}'.format(y_center, x_center - 1, 1)
                        else:
                                x_lowR = x
                                y_lowR = y
                                x_upR = x_center
                                y_upR = y_center - 1
                                print '{} {} {}'.format(y_center - 1, x_center - 1, 2)

                fill_carpet(y_topLeft + size/2, x_topLeft, size/2, y_lowL, x_lowL)
                fill_carpet(y_topLeft + size/2, x_topLeft + size/2, size/2, y_lowR, x_lowR)
                fill_carpet(y_topLeft, x_topLeft + size/2, size/2, y_upR, x_upR)
                fill_carpet(y_topLeft, x_topLeft, size/2, y_upL, x_upL)
```

Ryan Mysliwiec

YOU DO NOT NEED TO CHANGE THE CODE BELOW THIS LINE

Read input

```
data = [int(x) for x in sys.stdin.readline().split()]

carpet_dimen = data[0]
y = data[1]
x = data[2]
fill_carpet(0, 0, carpet_dimen, y, x)
```