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 2nx2n

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 2kx2k, with one square filled in. This board can be filled by (2k-1)/3 L-shaped carpet pieces.

Inductive Step: A grid with the size of 2k+1x2k+1 is made up of 4 2kx2k quadrants. The filled square is in one of those quadrants. When an L-shaped piece is placed at the center of the 2k+1x2k+1 board, each quadrant not containing the currently filled square are now “missing” a square as well, where each size is 2kx2k. 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 = n0 nlog24 = n1/2

a/bc = 4/20 = 4 > 1 => Case 1

ε = ½

n0 = n1/2 – ½

T(n) = θ(n1/2)

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)

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)

# 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)