Carson Sklare

2a. Code (incomplete, using c_hanoiH as recursive helper):

```
def c_hanoi(n):
    long(n,0,2)

def c_hanoiHr(n,a,b):
    if n == 0:
        return
    c = 3-a-b
    print(n)
    print('a: {}'.format(a))
    print('b: {}'.format(b))
    print('c: {}'.format(c))

    c_hanoiHr(n-1,a,c)
    print('Move disk from {} to {}'.format(a,b))
    c_hanoiHr(n-1,c,b)

def c_hanoiH(n,a,b):
    if n == 0:
        return
    c = 3-a-b


    if(b == 2):
        print('Move disk from {} to {}'.format(a,c))
        print('Move disk from {} to {}'.format(c,b))
        c_hanoiH(n-1,a,b)
    if(b == 1):
        print('Move disk from {} to {}'.format(b,a))
        c_hanoiH(n-1,b,c)
def c_hanoiH2(n,a,b):
    if n == 0:
        return
    c = 3-a-b

    c_hanoiH2(n-1,a,b)
    print('Move disk from {} to {}'.format(a,b))
    c_hanoiH2(n-1,a,c)
    print('Move disk from {} to {}'.format(c,b))
    c_hanoiH2(n-1,a,c)
```

Result (not correct):

```
>>> c_hanoi(4)
Move disk from 0 to 1
Move disk from 1 to 2
Move disk from 0 to 1
Move disk from 1 to 2
Move disk from 0 to 1
Move disk from 1 to 2
Move disk from 0 to 1
Move disk from 1 to 2
>>>
```

2b.

3.

a.
   i. t(n) = t(n/2) + t(n/2) = 2t(n/2) + n = 2(2t(n/4) + 2n/4)+n = 4t(n/4)+2n … 2^d +(n/2^d)+dn where d = log2n, 0+dn = n log n, same as mergesort
   ii. Wolfram alpha output:

t(n) =t(n/2) +t(n/2) + n, t(1) = 0

Extended Keyboard   Upload

Input:

$$t(n) = 2\,t\left(\frac{n}{2}\right) + n \quad | \quad t(1) = 0$$

Result:

$$\left\{t(n) = n + 2\,t\left(\frac{n}{2}\right),\ t(1) = 0\right\}$$

Recurrence equation solution:

$$t(n) = \frac{n\log(n)}{\log(2)}$$

   iii. O(n log n)

b. I would use merge sort to start splitting lists up, when I have the list of elements smaller than the right side that are added to the out array, I can get the number of inversions for that list by subtracting the length of the left side minus the iterator for left side

c. In my code it went to 1000000 before it took long

d. Code:

```python
def invert(lst):
    return inversions(lst)[1]
def inversions(lst):
    #merge and mergesort code from week2.py
    if len(lst) <= 1:
        return lst, 0
    mid = len(lst)//2
    L, l = inversions(lst[:mid])
    R, r = inversions(lst[mid:])

    i, j = 0, 0
    out = []
    count = 0 + l + r;

    while i < len(L) and j < len(R):
        if L[i] < R[j]:
            out.append(L[i])
            i += 1
        else:
            out.append(R[j])
            j += 1
            count+= (len(L)-i)
    if i == len(L):
        out += R[j:]
    else:
        out += L[i:]
    return out, count
```

Results:

```
>>> invert([4,3,5,2,1])
8
>>> invert([3,1,2])
2
>>> invert([1,2])
0
>>> invert([2,1])
1
```

e. Make length of the longlist n
   The inversion number will be around n log n