Sample Exam 1

(1) What is the running time of the code example below:

```
l = []
# l is an arraylist
for i in range(10):
    l.insert(i)
```

(2) Write an algorithm to reverse an array list in O(n) time.

(3) What is the running time of this algorithm. The running time of f(x,y) is O(ylogx).

```
sum = 0
for i in range(len(n)):
    sum += 1
for j in range(len(n*n)):
    f(sum,j)
```

(4) What does the following function do?

```
# Q is a Queue
def fun(q):
    stack = Stack()
    while(q.peek()): # while the queue is not empty
        s.push(q.dequeue())

    while(s.peek()): # while the stack is not empty
        q.enqueue(s.pop())

    return q
```

(5) Write a recursive method that returns the number of 1's in the binary representation of N. *Use the fact that this is equal to the number of 1's in the representation of N/2 (plus 1 if N is odd*

Solutions

(1) The running time is $O(N^2)$. This is because the for loop runs in O(N) but also inserting at the front of an arraylist also takes O(N). So the running time of the for loop is O(N*N) = $O(N^2)$

(2) The idea is we can use a stack to reverse the items and push them onto a new list. An alternative idea is to start from the end of the list and reverse it that way as well. Both run in O(N) time. The important thing is to remember to add at the end of the array list as that takes constant time and we start with a new arraylist of the same size which prevents from having to resize the list when we exceed the initial limit

```
def reverse_with_stack(a):
    stack = Stack()
    for i in range(len(a)):
        s.push(a[i])

    new_list = ArrayList(len(a))
    # Here we append at the end of the list
    while(s.peek()):
        new_list.append(s.pop())

    return new_list

def reverse(a):
    new_list = ArrayList(len(a))
    # In python the range function works as so range(start, end, increment)
    # The function also always stops 1 before then end so if we put -1 it will stop
        at 0
    for i in range(len(a)-1, -1, -1):
        new_list.append(a[i])

    return new_list
```

(3) The running time of this algorithm is $O(N^4logN)$ now we can see that the first for loop runs in O(N). We can also see that the second for loop runs in $O(N^2)$. With the rule of consecutive statements we would have $O(N + N^2) = O(N^2)$ but it's not that simple. In the second for loop we need to account for the running time of the function which is $O(ylogx)$. In this case we have f(sum,j) and sum = n and j = n*n. So we have f = $O(N^2logN)$ so the running time of the second for loop is $O(N^2 * N^2logN) = O(N^4logN)$. Back to the rule of consecutive statements we have $O(N + N^4logN) = O(N^4logN)$

(4) The function reverses the queue. It first pops all the elements and puts them into the queue. Since queue is FIFO then when we push elements back onto the Stack the elements will go in, in the same order they were popped. Since stacks always insert at the front we are pushing elements back on in reverse order.

(5) Here we create a function that recursively divides the number by 2 until we reach 0 and add 1 if the number is odd. Simply as the question was defined.

```
def num_ones(n):
    if n == 0:
        return 0
    else:
        return num_ones(n/2) + (n%2)
```