Alex Mosseri

1a) Theta(n)

1b) theta(n^2)

1c) theta(log(n))

2a)
3n^4 <= 3n^4 +8n^3- 3n <= 3n^4+ 8n^4
3n^4 <= 3n^4 +8n^3- 3n <= 11n^4

2b) sqrt(17n^2 +4n -7)
17n<= sqrt(17n^2 +4n -7) <= 17n +4n
17n<= sqrt(17n^2 +4n -7) <= 21n

2c)
We say that f(n) = O(g(n)) if there exists positive real constant C and a positive integer constant n(sub0) such that f(n) <= c(g(n)) for all n >= n(sub0). and then if g(n) = O(h(n)) then there exists positive real constant C and a positive integer constant n(sub0) such that g(n) <= c(h(n)) for all n >= n(sub0). So then at the conclusion you could say that f(n) = O(h(n)) if there exists positive real constant C and a positive integer constant n(sub0) such that f(n) <= c(h(n)) for all n >= n(sub0).

3) The running time of this is Theta(log(n)). This is it is proportional to the depth of the recursion tree. At each level, the square root of n is taken.

4)

1) The worst case run time for reverse1(last) is O(n^2) because it is going through a for loop n amount of times and then it is also inserting elements into the list at the 0 index which always pushes the data over one when adding another element.

2) The worst case run time for reverse2(lst) is O(n) because it is going through the list n amount of times and then it is appending to the end of the list which is a constant amortized