Code Analysis

Question 1

1. log2(n)
2. sqrt(n)
3. 100n
4. nlog2(n),
5. n^3/2
6. 5n^2+10n
7. 4n^2
8. n^3 + 5n – 100
9. 3.2^n
10. .0001n!

Question 2

**Segment 1:** T(n)=2 which is O(1). Despite the len function, as an array the length is precalculated and takes constant time to retrieve.

**Segment 2:** T(n)=nlog2(n) which is O(nlog2(n)) as this is the the big-O of merge sort.

**Segment 3:** T(n)=4 I believe, but is O(1) as it takes constant time to find the middle of the array and to access these values. Even with the math involved, this is constant time regardless of the input.

**Segment 4:** T(n)=n+2 I believe, which is O(n) as we must traverse the array to calculate it’s sum.

T(n) = 1+ nlog(n) + 1 + n or nlog2(n) +n + 2

Meaning that O(n) = nlog2(n) as this is the highest order term

* I’ve added I believe to growth functions that include static lines of code that will be run regardless and didn’t include if statements though that could be a mistake. I think my concepts are solid here though with the Big-O complexity