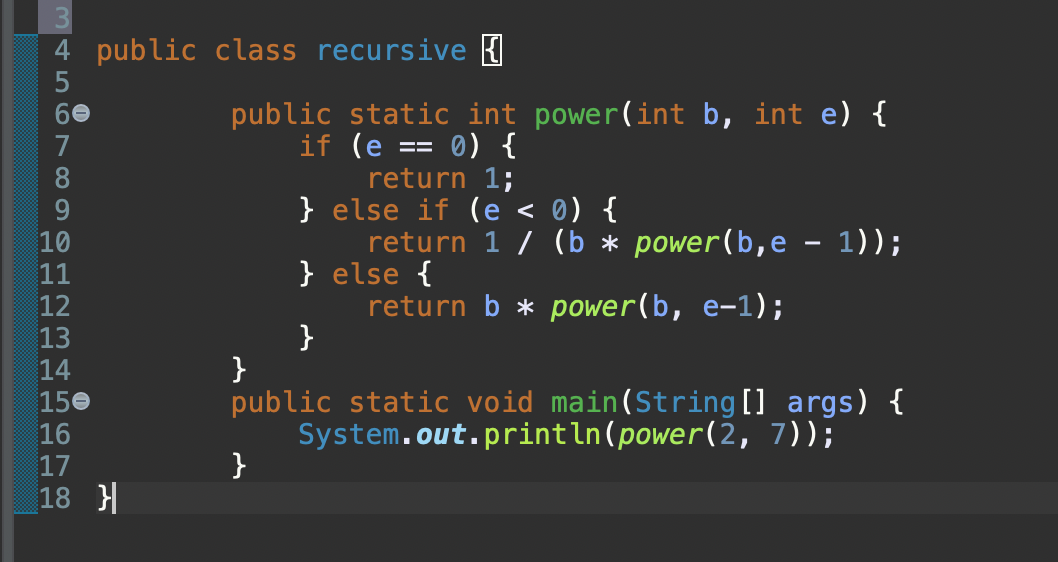
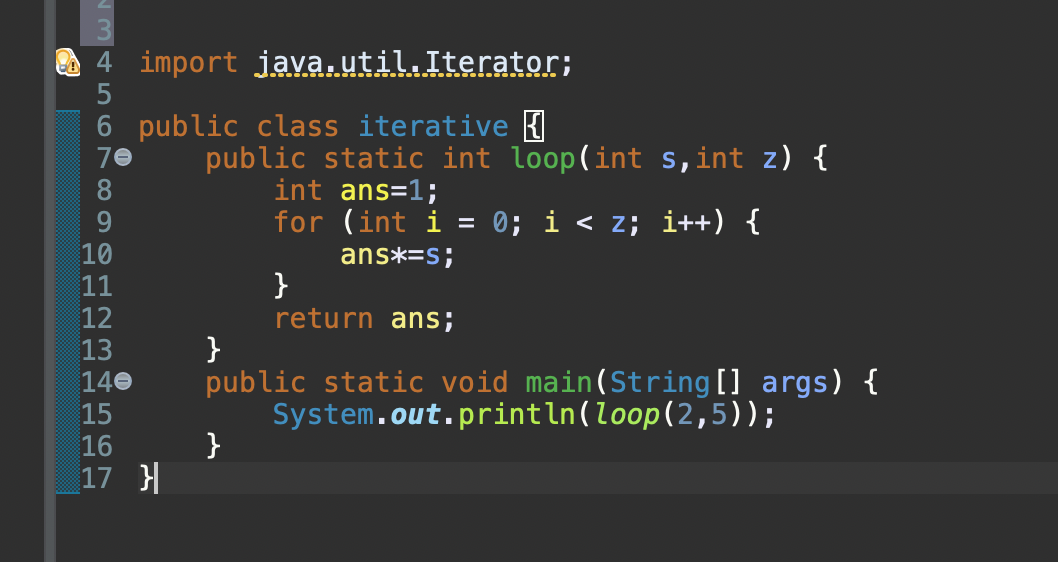
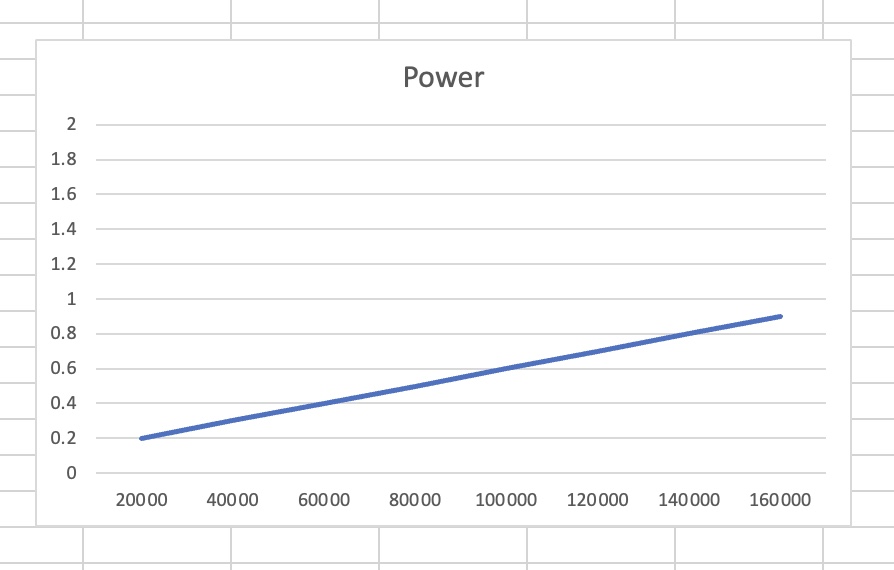
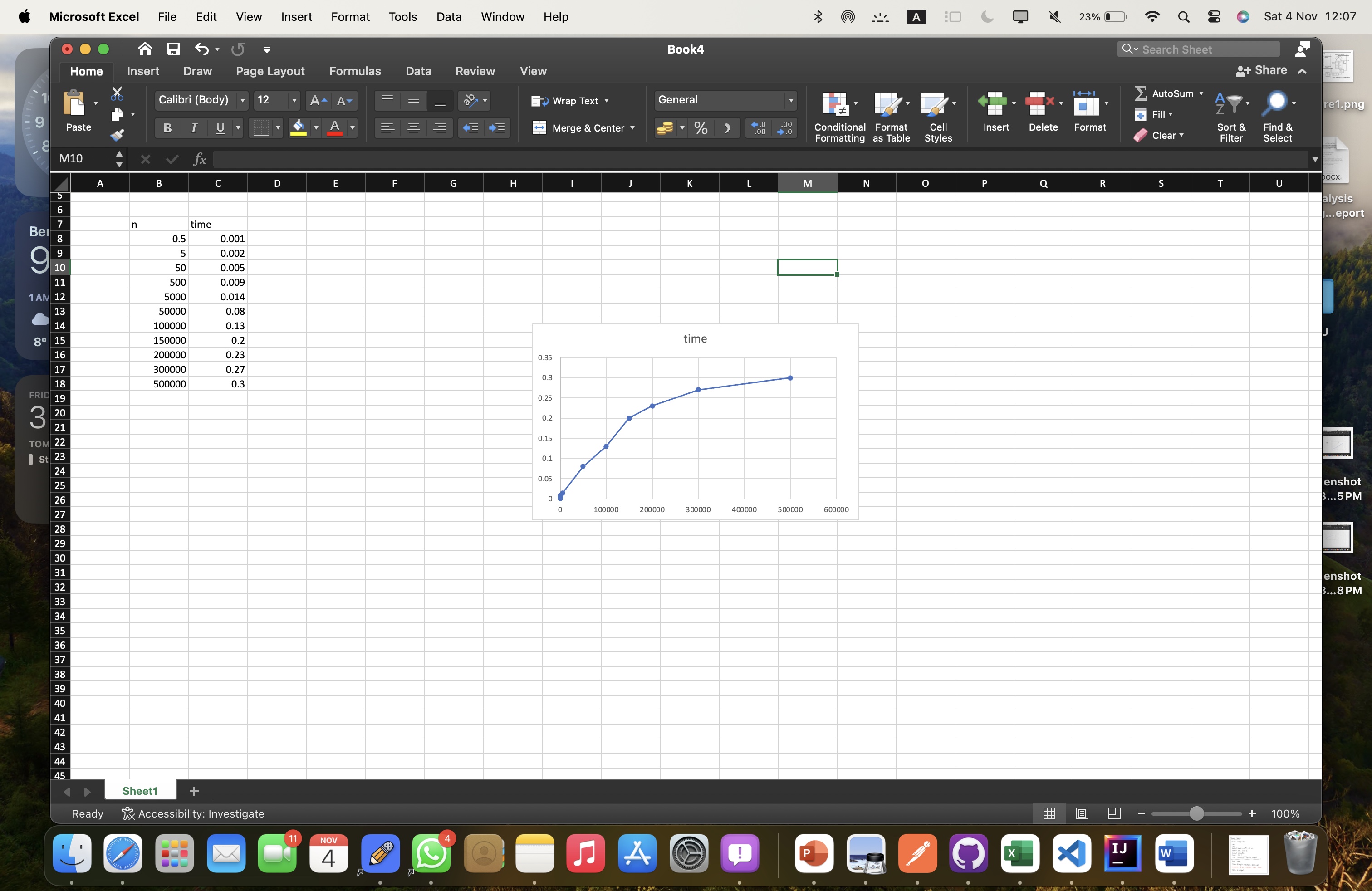
Analysis assignment. Alyeldin Zwail 52-1198

Question 1:

(a).

(b).

* + For the naive iterative method, the time complexity is O(n).
  + For the divide-and-conquer approach, analyze it as mentioned earlier and find that it's O(log n).

(c).

(d). Yes it does.

Question 2:

(a).

(b).

1.In the context of the MergeSort algorithm with an input array 'n', where a = 2, b = 2, and f(n) = O(n), we apply the Master Theorem. We compare the function f(n) with n^log\_b(a): f(n) is O(n), and n^log\_b(a) is n^log\_2(2), which simplifies to n. Comparing these two functions, since f(n) is O(n), it falls under Case 2 of the Master Theorem. Case 2 states that if f(n) is Θ(n^log\_b(a)), where log\_b(a) is less than 1, then the time complexity is given by T(n) = Θ(n^log\_b(a) \* log(n)).

2. Regarding the Binary Search algorithm for an input array 'n' and target element 'x', with a = 1, b = 2, and f(n) = O(1), the time complexity is determined to be Θ(log(n)) according to the Master Theorem. Since a = 1, b = 2, and f(n) = O(1), it directly falls into Case 2. Thus, the solution for the Binary Search recurrence relation is T\_binary(n) = Θ(log(n)). This means that the Binary Search algorithm has a logarithmic time complexity concerning the size of the input array.

3.When analyzing the time complexity of the 'findPairs' function for an input array 'n' and a target sum 'Sum,' it is determined that the greater of the two recurrence relations will be O(nlogn). This is because the function is invoking other methods that have this time complexity.

(c).