Jason Green
SER 222 – Module 5 – Elementary Sorts

Prior to running sorting algorithms in an IDE, the below are my hypothesis for each data set on two sorting algorithms; **Insertion Sort** and **Shell Sort**.

1. Insertion Sort:
    a. Binary:
        i. Hypothesis: O(n)
        Rationale: Due to data being sorted the algorithm can step through the data in constant time giving O(n)
    b. Half:
        i. Hypothesis: O(n)
        Rationale: Here, data is again sorted even though the data halfway through steps n+1 for every n/2 widths. The algorithm can step through the data in constant time giving O(n)
    c. Random:
        i. Hypothesis: O($2^n$)
        Rationale: Data is not sorted. After halfway through the data set, there are random values that the algorithm must check against. Since there is a swap of values for every n data set, then Big-O notation is O($2^n$)
2. Shell Sort:
    a. Binary:
        i. Hypothesis: O(n)
        Rationale: The data is sorted; however the width of the "shell" is 1/3 the length of the data set. Even so, Big-O notation is O(n) since the width is comparing values that are already sorted. The algorithm can step through the data in constant time.
    b. Half:
        i. Hypothesis: O(n)
        Rationale: As in the binary data set, the data is sorted with n+1 for every n/2 width of the data set. As the shell sort runs, the width decreases, but still finds the data set to be sorted. Big-O notation is still O(n)
    c. Random;
        i. Hypothesis: O($2^n$)
        Rationale: Data is randomized after ½ data length. Shell sort will have to swap values while the width of the shell decreases. This swapping gives O($2^n$)