Problem 1. Assume that you have a list of size \( n \) where every value occurs exactly twice.

(a) What is the best-case number of comparisons for Insertion Sort with a sentinel.

(b) What is the worst-case number of comparisons for Insertion Sort with a sentinel.

(c) What is the average-case number of comparisons for Insertion Sort with a sentinel.

Problem 2. Let \( A \) be an array of \( n \) distinct values.

(a) Give a quadratic-time algorithm based on Insertion Sort without a sentinel to create an array \( \text{WHEREIS} \) of \( n \) values so that \( \text{WHEREIS}[i] \) is the index of the \( i \)th smallest element of \( A \). For example, if \( A \) is

\[
(40, 80, 30, 60, 10, 70, 20, 50)
\]

then \( \text{WHEREIS} \) should be

\[
(5, 7, 3, 1, 8, 4, 6, 2).
\]

You may not modify \( A \). \( \text{WHEREIS} \) can only hold “index values” not “array values” from \( A \). Technically, this means that a value in \( \text{WHEREIS} \) may only use about \( \lg n \) bits. Other than that, you may only use a constant amount of extra memory.

(b) Starting from your array \( \text{WHEREIS} \) from Part(a), give a linear-time algorithm to modify \( \text{WHEREIS} \) so that \( \text{WHEREIS}[i] \) is the rank of the \( i \)th element of \( A \). For example, if \( A \) is

\[
(40, 80, 30, 60, 10, 70, 20, 50)
\]

then \( \text{WHEREIS} \) should now be

\[
(4, 8, 3, 6, 1, 7, 2, 5).
\]

You may not modify \( A \). You may only use an extra \( n \) bits along with a constant amount of extra memory. (More formally, you may use \( n + O(\log n) \) extra bits of memory.)