Experimenting with Binary Search Trees & Selection Sort

In this project you will create a number of Binary Search Trees from a set of random data. Each
tree will differ in how it is constructed. You will investigate certain properties for each tree and report
on your findings. You will begin by creating an array of \( n \) random elements. For the purpose of this
assignment we will consider \( n \) to be 55. The random elements will consist of integers in the range \([0, 999]\).

First create a binary search tree directly from the array. The first element in the array will be
inserted first, becoming the root of the tree. The second element in the array will be inserted second,
and so on.

You will repeat the tree building process for the following nine conditions:

1. The element with the largest value in the array is inserted first.
2. The element with the smallest value in the array is inserted first.
3. The element that is equal to the median value in the array is inserted first. To find the
median make a copy of the input array, sort it, and then take then \( n / 2 \) element.
4. Trees where the first element inserted is the median of the first \( p \) elements, where \( p = 5, 9, 15, 17, 21 \). To find the median, copy the first \( p \) elements, sort them, and take the \( p / 2 \) element.
5. The last element in the array is inserted first and the rest of the elements are inserted in
reverse order (e.g. from \( n-1 \) down to \( 0 \)).

Using the pseudo code given in class, implement an iterative Selection Sort. Implement two
functions named \( \text{Index}(A, n) \) and \( \text{SelectionSort}(A, n) \). The sorting to find the median in parts 3 and 4
must use your implementation of Selection Sort. You must implement the Selection Sort algorithm
given in class for any credit. Within the Selection Sort algorithm, calculate the total number of \textbf{key comparisons}
required to sort the input array.

For all Binary Search Trees created, calculate the following values:

1. The number of \textbf{key comparisons} required to build the tree.
2. The sum of the node path lengths for all nodes.
3. The average node path length of the tree.
4. The maximum node path length of the tree.

For all of the trees print out the input sequence of values and the values calculated at the four
previous steps. Also, for each tree apply a \textbf{recursive} in-order traversal of the tree and print the output.
For the trees rooted at the first element in the array, the maximum element in the array, and the median
element of the array draw the trees by hand.

To represent the binary tree use a linked structure containing nodes with the fields \textbf{LLink},
\textbf{value}, and \textbf{RLink}. You may also need the fields \textbf{Lvalue} and \textbf{Rvalue}, which can be printed to aid in
drawing the tree.