1. (10 points) In the closest-pair problem, let \( P \) be a point set with \( n \) points. Let \( X \) be the same point set sorted by \( x \)-coordinates. For points with the same \( x \)-coordinate, they are sorted by \( y \)-coordinates. Let \( Y \) be the same point set sorted by \( y \)-coordinates. For points with the same \( y \)-coordinate, they are sorted by \( x \)-coordinates. Recall that the closest-pair algorithm uses an imaginary vertical line to bisect \( X \) into \( X_L \) and \( X_R \), which is easy to implement. As a result, \( Y \) is also partitioned into \( Y_L \) and \( Y_R \), where \( Y_L \) and \( Y_R \) are the same point sets as \( X_L \) and \( X_R \), respectively, sorted by \( y \)-coordinates. Describe in words a \( O(n) \)-time algorithm to create a partition of \( Y \) into \( Y_L \) and \( Y_R \).

2. (10 points) Use the divide-and-conquer approach to write an algorithm that finds the largest item in a list of \( n \) elements. Analyze your algorithm and show the time complexity in big-O notation.

You may call your algorithm \( \text{SelectMax}(A) \), which is a function that returns the maximum value in \( A \). Don’t forget to start your algorithm with an “if” statement that specifies the base case for the recursive algorithm. Once you have the algorithm, the time complexity should be represented by a recurrence relation. Solving the recurrence relation gives you the time complexity in big-O.