1. Using the pseudocode algorithm for selection sort provided in the lecture notes, do a detailed analysis ($T(n)$ function) for array comparisons (i.e., just line 5) and another detailed analysis ($T(n)$ function) for moves (i.e., just line 6). Note that one swap is considered three moves.

2. Using the pseudocode algorithm for insertion sort provided in the lecture notes, do a detailed analysis ($T(n)$ function) for array comparisons and another detailed analysis ($T(n)$ function) for moves. Note that:
   - the best case is when the array is in ascending order and worst case in descending order
   - your two functions must consider both best and worst cases
   - you only need to count the number of comparisons $\text{array}[j-1] > \text{val}$ (line 4)
   - moves occur on line 2, 5, and 7 (where each line incurs a cost of one time unit)
3. Using the pseudocode algorithm for bubblesort provided in the lecture notes, do a detailed analysis ($T(n)$ function) for array comparisons (line 5) and another detailed analysis ($T(n)$ function) for moves (line 6). Note that the best case for bubble sort is when the array is in ascending order and the worst case is when the array is in descending order.

4. Provide overloaded versions of the increment operator ($++$) for the Rectangle class from the lecture notes. Note that there are two versions of increment: a prefix version (e.g., $++x$) and a postfix version (e.g., $x++$). The prefix version increments $x$ then returns the incremented value. The postfix version makes a copy of $x$, then increments $x$, then returns the copy. The two versions of the operator would have the following signatures for Rectangle:

- \[ \text{Rectangle} & \text{Rectangle::operator++()}; \] \hspace{1cm} \text{... prefix version}
- \[ \text{Rectangle Rectangle::operator++(int);} \] \hspace{1cm} \text{... postfix version}

Note that the int argument in the postfix version is just there to distinguish between the two versions. For Rectangle, increment should add one to both the width and length.