1. (15 points) Given the definition of a class template, write the C++ code for a specific overloaded operator.

2. (15 points) Write the C++ code for a specific function template.

3. (4 points) Explain the assumptions that you must make about the parameterized class to make your function template work in problem #2.

4. (2 points each) For each of three provided diagrams of functions $f(n)$ and $T(n)$, indicate whether or not the function $T(n)$ is $O(f(n))$.

5. (15 points) Given a specific C++ function, prove that it has a particular big-O time complexity.

6. (1 point each) Ten true-false questions concerning time complexity and sorting algorithms.

7. (4 points) Illustrate the steps of a particular sorting algorithm when applied to a specific unsorted list of integer values. Possibilities include:
   - Bubble Sort: Specify what the list will look like after each pass of the Bubble Sort.
   - Selection Sort: Specify what the list will look like after each pass of the Selection Sort.
   - Insertion Sort: Specify what the list will look like after each pass of the Insertion Sort.
   - Shell Sort: Specify what the list will look like after each pass of the Shell Sort, using a specific set of increments.
   - Quick Sort: Specify what this list will look like after the first pass of a Quick Sort, using first-element pivoting.
   - Merge Sort: Specify what the list will look like after each pass of the Merge Sort.
   - Radix Sort: Specify what the list will look like after each pass of the Radix Sort, using a specific bin-loading strategy.

8. (4 points) Another problem like #7, but using a different sorting algorithm.

9. (4 points each) Given a specific sequence of integer values, draw the tree that results from inserting these numbers in sequence into:
   - (a) a binary search tree.
   - (b) a maximum heap, assuming that a complete binary tree is always maintained.
   - (c) an AVL tree.

10. (15 points) Given the class definition for a template for linked binary trees, write two new member functions for this class, a protected function that performs a specific recursive traversal of the tree and a public function that kicks off the recursive call to that protected function.