**Scoring Rubric for Project 6: Insertion Sorting a Linked List**

*Due 9/4/2020 @ 12:30 pm*

|  |
| --- |
| Student Name: Nate |

|  |  |  |
| --- | --- | --- |
|  | **Score** | **Maximum** |
| **Execution (50 pts):** | | |
| Program compiles without errors (warnings are okay) | 50 | **50** |
| **Implementation (45 pts):** | | |
| Implements the LinkedList class to be a friend of the Node class and contain only one pointer “head” | 5 | **5** |
| Implements a member function of LinkedList to do the insertion sort: implementation designs can vary, but some operations should be encapsulated in helper functions and it should be quadratic time | 13 | **15** |
| Implements the “big three”: copy constructor, copy assignment operator, and destructor for the LinkedList class | 10  The copy constructor is not right. \*head = \*(list.head); should be Node \*head= new Node(head->value, head->next) | **15** |
| Plots the execution time vs. N for the vector and linked list | 5 | **5** |
| Describes the similarity or difference observed in the performance of the InsertionSort algorithm for a vector vs. Linked List | 5  Your description is not quite correct. The binary search is O(logn) and the insertion sort for the vector becomes O(logn\*n^2) but the insertion sort in Linkedlist is still involved the search for the right location which makes it O(n^2) | **5** |
| **Style (5 pts):** | | |
| The driver and functions are easy to follow based on the use of comments | 3 | **3** |
| Easily identifiable variable names | 2 | **2** |
| **Total (100 pts):** | 92 | **100** |

Notes: