**Scoring Rubric for Project 2 : Mergesort**

*Due 9/19/2019 @ 5 pm*

|  |
| --- |
| Student Name: Sarah Ruth Nicholls |

|  |  |  |
| --- | --- | --- |
|  | **Score** | **Maximum** |
| **Execution (50 pts):** | | |
| Program compiles without errors (warnings are okay) | 50 | **50** |
| **Implementation (40 pts):** | | |
| Uses function declarations as provided | 5 | **5** |
| Main function includes at least one unit test for MergeSortedLists (can use assert or printed output) | 5 | **5** |
| MergeSortedLists works for both even and odd sized input | 5 | **5** |
| MergeSortedLists works for both left and right lists emptying first | 5 | **5** |
| Implements MergeSort base case correctly (may be implicit) | 5 | **5** |
| MergeSort recursive case follows pseudocode: two recursive calls followed by call to MergeSortedLists (may have indexing bugs) | 5 | **5** |
| MergeSort passes 5 different unit tests (2 points each) | 10 | **10** |
| **Style (10 pts):** | | |
| The driver and functions are easy to follow based on the use of comments | 6 | **6** |
| Easily identifiable variable names | 4 | **4** |
| **Total (100 pts):** | 100 | **100** |

Notes:

Great work!

When you’re checking the unit test for mergeSortedLists, you really should have assert(test.at(i-1) <= test.at(i)) instead of assert(test.at(i-1) <= testTemp.at(i)). Even though testTemp does ultimately contain the sorted list, the list we’re actually interested in is test, which will also end up being sorted.