**Scoring Rubric for Project 2 : Mergesort**

*Due 9/19/2019 @ 5 pm*

|  |
| --- |
| Student Name: Zexuan Huang |

|  |  |  |
| --- | --- | --- |
|  | **Score** | **Maximum** |
| **Execution (50 pts):** | | |
| Program compiles without errors (warnings are okay) | 50 | **50** |
| **Implementation (40 pts):** | | |
| Uses function declarations as provided | 2.5 | **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):** | 97.5 | **100** |

Notes:

You should not be allocating any extra arrays in mergeSortedLists, but do all your work in the tmp vector that is passed in with the function call.

If you want to test for equality between “test” and “expected” using assert, you should start the for loop at 0 and have the assert on “test.at(i) == expected.at(i)”. The reason for starting the loop at 1 and checking i-1 is to make sure the previous value in the vector is smaller than the next.