CSCI 1933 Project 3 Rubric
Spring 2021

Totals

100 points possible for all students.

Global Penalties

- **-5 points**: Within their code, they use `==` rather than `.compareTo()` in methods where they are comparing `T` data. This will require a bit of code inspection.

  Note: the unit tests were written in a way where the tests passed even if they used `==`. This was a mistake and should not be a valid way of comparing `T` data.
- **-10 points**: No README included.
- **-5 points**: Does not follow project structure as laid out by write-up.
- **-10 points**: Late by 1 day
- **-20 points**: Late by 2 days
- **-10 points**: Trivial compilation errors. This includes anything that can be fixed quickly (e.g., parenthesis, incorrect class names, missing semi colons).
- **-40 points**: Nontrivial compilation errors. This covers all other cases when the code is unable to compile on its own for reasons unrelated to the linker or java version.
- **No Credit**: Late by more than 48 hours

Style (4 points)

- **2 points**: The code contains helpful comments, especially for more complex operations.
- **2 points**: The code is not needlessly complicated, and is structured in a way that makes it easy to read.

Note: If the student does not use modular arithmetic for rotating the list (i.e., the `rotate()` function), we should leave a comment explaining why that would be beneficial to the efficiently of their code. For example, if they were rotating a list of length 5 by 1,000,000,000 rotations, then their code should not brute force this and rotate each element 1,000,000,000 times. Instead, they should have rotated each element equal to the number of rotations modded by the length of the list.

ArrayList (40 points)

- **37 points:** Unit tests. Full points for passing a test, no points for failing a test.

- **3 points:** Properly using isSorted to increase the efficiency of the methods.
  - 1 point for using isSorted in indexOf method to increase efficiency of method.
  - 1 point for using isSorted in sort method to increase efficiency of method.
  - 1 point for using isSorted in equalTo method to increase efficiency of method.
- **PENALTY:** (-2) For using an intermediate data structure to reverse the list (e.g., initializing a second array and copying over the elements from the old array to the new one).
- **PENALTY:** (-5) For using an intermediate data structure to rotate the list (e.g., initializing a second array and copying over the elements from the old array to the new one).

LinkedList (40 points)

- **37 points:** Unit tests. Full points for passing a test, no points for failing a test.
- **3 points:** Properly using isSorted to increase the efficiency of the methods.
  - 1 point for using isSorted in indexOf method to increase efficiency of method.
  - 1 point for using isSorted in sort method to increase efficiency of method.
  - 1 point for using isSorted in equalTo method to increase efficiency of method.
- **PENALTY:** (-2) For using an array when sorting.
- **PENALTY:** (-2) For using an intermediate data structure to reverse the list (e.g., using an array as an intermediate data structure).
- **PENALTY:** (-5) For using an intermediate data structure to rotate the list (e.g., using an array as an intermediate data structure).

Note: The Reverse method for a linked list has been gone over in lecture. Do not be surprised if you see a lot of students with the same/similar solution for this method.

Runtime Analysis (16 points)

Only 4 of the methods will be graded for runtime complexity. The grading criteria for all methods will be the same (4 points per method): 1 point for the correct ArrayList implementation complexity, 1 point for the correct LinkedList implementation complexity, 1 points for the correct ArrayList explination, 1 points for the correct LinkedList explanation. You will need to verify that their code has the time complexity that they listed on the document.

- **4 points:** add(T element) method.
- **4 points:** rotate(int n) method.
- **4 points:** merge(List<T>otherList) method. The students were told to ignore the time complexity of first sorting the two lists.
- **4 points:** reverse() method.