John Rudolph

JAR

CS1350

Project #2

Quick Sort and Insertion Sort

due date: 11/8/2020

CS1350 Programming Project 2 Sorts Fall 2020

Due November 8, 2020 Mansfield, J

Design, test, implement and test a C++ program that uses a linked list data structure that first builds the linked list then uses both a quick sort (one program) and an insertion sort (second program) to sort the integers. As the sorts are executing, count the number of comparisons made and the number of data moves (how many pointer assignments required). Your program will be tested on three different data sets – random, nearly sorted, completely unsorts (sorted in reverse of desired sort.)

Sort so that the smallest value is in the ‘lowest’ position.

You should plan this program well.

Make sure to include all of your documentation for:

a) planning – algorithms

b) test and evaluation design – a plan to test each part of your program, along with the test data that will be used and how you will determine if test passed or failed

c) reflection – separate document, example available (25% of grade)

**Documentation – submit in appropriate location on BB:**

Turn in all required documentation:

• Cover page

• Copy of assignment

• Design documents – including all diagrams, algorithms, testing information, etc.

• Reflection

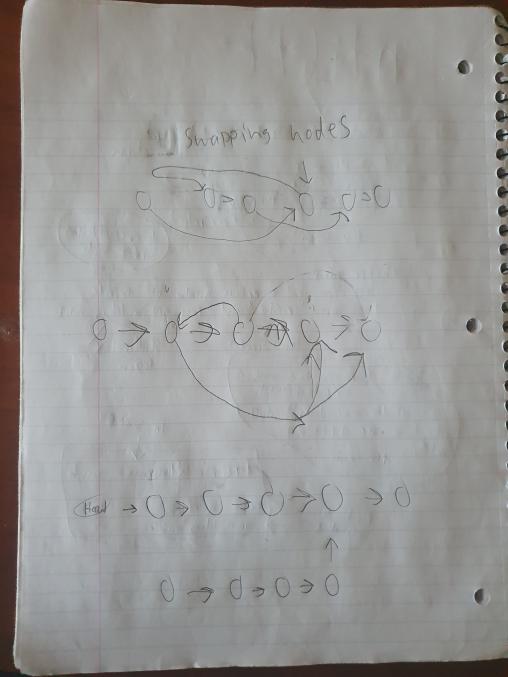
**Code – submit in appropriate location on BB:**

* One compressed folder:
* named yourinits.sorts – use your initials.sorts
* all code and input/output files need to be in this folder
* all files need to have comments at the top:
  + //your name
  + //program name and portion of the program this file covers
  + //due date of the program
* all files need to have appropriate names, recommended naming: xyz.quick.cpp and xyz.insert.cpp

Due Date: 11/8/20

Algorithm:

Use this area to put your algorithm



The hardest part of my algorithm is swapping nodes. Everything else I can reference in the source code provided from earlier lessons.

I will test my functions by filling my linked lists with valid data and running my quick sort and insertion sort on those lists.

Project Summary:

We were tasked to create two sorting algorithms to sort singly linked lists, using both quick sort and insertion sort.

Challenges:

The part of the project I found most challenging was having to convert my code designed for a doubly linked list into code that could support a singly linked list.

Solutions (Mitigation):

I had to reference the source code multiple times, as well as looking online for help. In future projects I will not waste as much time trying to convert code. Instead, I should have scraped my old code and started over.