1. Study Nguyen, Chapter 4.2 subsections Implementation of HeapSorter, Implementation of SelectSorter and InsertSorter, and Section 4.3.

2. Do Nguyen Exercise 4–7(a) to rewrite the `siftDown()` function in `Heapifier.hpp` without recursion.

   `Heapifier.hpp` is in the `ASorter` project. Note the optimization requirements in the problem statement.

   Hint: Here are two lines of code to get you started. Terminate your `while` loop on `!done` where `done` is a boolean variable, and set `done` accordingly in the body of your loop. Do not use the C++ `break` statement.

   ```
   int child = 2 * i - lo + 1; // Index of left child.
   bool done = hi < child;
   ```

   First, test your program with the `SortInt` project, which sorts integers.

   Then, test your program with the `SortCompAsgn` project, which contains the twelve files

   ```
   d0500.txt
   d1000.txt
   d1500.txt
   d2000.txt
   d2500.txt
   d3000.txt
   d3500.txt
   d4000.txt
   d4500.txt
   d5000.txt
   d5500.txt
   d6000.txt
   ```

   with 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, and 6000 random real values respectively. Remember to change the working directory in CLion for the `SortCompAsgn` project.

   Hand in the following three files:

   ```
   MergeSorter.hpp
   Heapifier.hpp
   InsertSorter.hpp
   ```

   If you do not hand in three files the project will not compile and you will receive automatic 20% credit. Hand in the three files electronically with your two-digit course ID number prefixed to each file. For example, if your two-digit ID number is 87, hand in the files named

   ```
   87MergeSorter.hpp
   87Heapifier.hpp
   87InsertSorter.hpp
   ```

   Fix any bugs that may have been in `MergeSorter.hpp` from Assignment 6 and `InsertSorter.hpp` from Assignment 7. This code will be the basis for your data collection for the paper in Assignment 9.