**CSE 12 SP21 Final Assessment**

Due Friday, June 11, 2021, 11:59pm **(no late assignments accepted)**

In this assignment, you are asked to implement a polymorphic generic container array-based ***heap data structure***. We will not provide any starter code for this assignment, but we will supply a sample executable, so it is your responsibility to use your knowledge from the previous assignments to have a working Driver executable using your class definition for the Heap class. Though we suggested some debug messages in the Driver executable, you have full control over your debug messages. Also, your

Method signatures are up to you and your imagination! Well... as long as:

1. Your output matches the sample executable (that will be posted shortly. Thank you for your patience).

2. All methods not called from the Driver are private.

3. You follow CSE 12 course style and commenting guidelines.

4. You practice appropriate layering.

5. Your array-based heap is a polymorphic generic container that accepts UCSDStudents as used in hw6, hw7 or hw8.

6. Your program runs from a bash prompt with: ./Driver

7. Using a Makefile you provide, the following Makefile commands function as expected:

make new

make Driver

8. Your code needs to be in ~/hw10. (case must match exactly!)

9. Your source files need to be called (case must match exactly!)

Java: Driver.java, Heap.java

* Additionally, you can also use any Java file from your previous assignments if you want (Base.java, MyLib.java, for example).

10. The data stored in your heap must use an array of generic items. Use of an ArrayList, Vector or any other data structure to store data in your heap is ***not allowed.***

11. Your heap code should display the contents of the heap to the user, catastrophic error messages, and debug messages. All other output comes from your Driver code. Inserting into a full heap and removing from an empty heap shall not generate output from your heap code but shall be detected by your Driver code.

12. ***ABSOLUTELY NO LATE SUBMISSIONS ARE ALLOWED!*** Start early, start often, and ***submit often***. Re-submission of your code before the due date is allowed. DO NOT wait until the last minute to submit your code. You should turn some code in before the due date, otherwise you risk getting a 0 on the final. There is no extra credit awarded for this final assignment for early submission.

# 

# **What is a Heap Data Structure?**

A heap data structure is a specialized binary tree array-based data structure with the properties that the tree is complete (which means every level is completely filled except possibly the leaf nodes and all the leaf nodes are as far left as possible) and parent nodes have higher priority than the child nodes. **For this implementation, items are removed from the heap in alphabetic order.** Therefore the name of a UCSDStudent earlier in the alphabet is higher in importance, and hence is higher in the Tree.

For example:

**A**

**/ \**

**B C**

**/ \ /**

**D E F**

This Heap as an array (which is what you will be implementing, an ARRAY BASED Heap) would look as the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 |
| Index | A | B | C | D | E | F |

For more information about Heaps and the equation please refer to **Week 8 Lecture 1** handout and to the following video starting at 4:20 [**Stacks, Queues, and Heaps**](https://www.youtube.com/watch?v=GYAPIxiTjww)**.** Also, feel free to do your own research about how heaps work!

# **How to start:**

First, we suggest you copy over a past Driver.java file from homework 6, 7 or 8. The Driver functionality needs to insert and remove UCSDStudents. It should also have functionality to display the heap to the user. You should already have a ***isLessThan*** method written for the UCSDStudent class (this method will be very important to compare item importance in the insert and remove algorithms, since you need to maintain the heap property that the child will be greater in importance than its parents). **Remember, in prior assignments with these methods A was less than Z, but now A is more important than Z.**

Next, you need to write your class definition for the heap container using an array to hold the data. You can use the below for guidance for which helper methods to implement besides the constructor, destructor, write, insert and remove methods. It should all go inside the Heap.java file.

# **Restrictions on Implementation**

Since this program counts for your final exam, tutor assistance is similar to a final exam in that you can use Piazza only to ask clarifying questions. Just as you can’t ask the tutor to debug your exam, asking for debugging assistance on Piazza is not allowed. Any requests for assistance with debugging will be ***ignored and result in a project deduction.***

# **Suggested Methods to Implement:**

--- Use of UCSDStudent in the below suggestions is as an example test object for your Driver code. As you did in hw6, hw7 or hw8, your Heap should be able to store any derived class from Base, any derived class from Object, or any type used as an instantiation argument.

* **Heap**: this is the constructor. It should initialize all your instance variables that you might want to use. It should take a size, which will be the size of your array used to store the data.
* **jettisonHeap**: this is the jettison method. It should jettison all associated memory to account for memory usage.
* **insert**: inserts a UCSDStudent object. Each element is inserted in the array at the “final leaf,” and then reheap-up is performed to find its correct location.
* **remove**: removes the most important UCSDStudent object (stored at the root). The root element is replaced by the element at the “final leaf,” and then reheap-down is performed to find its correct location.
* **write**: displays the Heap and all elements to the user.
* **reheapUp** (called from *insert*): starting at the “final leaf,” this method should move the inserted element up the tree until it satisfies the “heap-order,” swapping places with its parent until it is less important than its parent and more important than its children.
* **reheapDown** (called from *remove*): starting at the root, this method should move the former “final leaf” element down the tree until it satisfies the “heap-order,” swapping places with its most important child until it is less important than its parent and more important than its children.
* To help implement good layering decisions, you may want to consider having helper methods. You can use the list below for ideas. Each item below could be implemented as a single function or multiple single line functions. Keeping each function as simple as possible will help reduce the implementation and debugging time:
  + get the parent, right child or left child of an item at a particular index
  + get the index of the parent, right child or left child of an item at a particular index
  + get an element at a particular index
  + checks if an element at an index has a left or right child
  + checks if an element at an index is valid or empty
  + swaps any two elements
  + checks if is the heap empty or full
* Please return to the Week 10 Lecture Two handout for more suggestions.

# **Advice:**

* Using the alphabet and reverse alphabet will be helpful in testing.
* Draw a heap on paper and verify your heap’s Driver display.
* Run the sample executable using the following command in your cs12sp21b account: “~/../public/hw10/Driver”. You should match this executable as much as possible.
* ***Make sure to test your code extensively!***

# 

# **What files do you need to write and turn-in?**

* Driver.java
* Heap.java
* Makefile
* All files ending with .java in your hw10 directory will be collected.

# **Turnin Procedure**

* Go to your hw10 directory: “cd ~/hw10”
* Run the turnin script in your hw10 directory: “turnin”
* Enter your Autograder Login information, and follow the instructions on the terminal. Choose “Final Assessment” when you are prompted to “Select an assignment”.
* Please refer to the turnin procedure on the CSE 12 Course Website for more information: [Turnin Procedure](http://ieng6.ucsd.edu/~cs12x/turnin.html)