Laboratory 11: Cover Sheet

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Place a check mark in the *Assigned* column next to the exercises your instructor has assigned to you. Attach this cover sheet to the front of the packet of materials you submit following the laboratory.

Activities	**Assigned:** Check or list exercise numbers	**Completed**
Implementation Testing	✓	
Programming Exercise 1		
Programming Exercise 2		
Programming Exercise 3		
Analysis Exercise 1		
Analysis Exercise 2		
	Total	

Laboratory 11: Implementation Testing

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Check with your instructor whether you are to complete this exercise prior to your lab period or during lab.

Test Plan 11-1 (Heap ADT operations)			
Test case	**Commands**	**Expected result**	**Checked**

Laboratory 11: Programming Exercise 1

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Test Plan 11-2 (Priority Queue simulation results)		
Time (minutes)	**Longest wait for any low priority (0) task**	**Longest wait for any high priority (1) task**
10		
30		
60		

Question 1: Is your priority queue task scheduler unfair—that is, given two tasks T_1 and T_2 of the same priority, where task T_1 is enqueued at time N and task T_2 is enqueued at time $N + i$ ($i > 0$), is task T_2 ever dequeued before task T_1?

My PrioirityQueue is currently implemented in an unfair manner. Items of similar priority are able to "cut" in line depending on how the insertion goes and what higher priority items they fall in behind.

Question 2: If so, how can you eliminate this problem and make your task scheduler fair?

A Comparator class could be implemented to compare both task priorities and arrival times. This would be a simple fix considering how our Heap ADT utilizes this use of a class as a function. The PriorityQueue could also be implemented with an inner class that contains both the item/task to be inserted as well as the arrival time (a second priority) to enhance this functionality.

Laboratory 11: Programming Exercise 2

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Test Plan 11-3 (heapSort operation)			
Test case	**Array**	**Expected result**	**Checked**

Laboratory 11: Programming Exercise 3

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Test Plan 11-4 (The writeLevels operation)			
Test case	**Commands**	**Expected result**	**Checked**

Laboratory 11: Analysis Exercise 1

Name: Terence Henriod Date: November 14, 2013

Section: 1001

You can use a heap—or a priority queue (Programming Exercise 1)—to implement both a first-in, first-out (FIFO) queue and a stack. The trick is to use the order in which data items arrive as the basis for determining the data items' priority values.

Part A

How would you assign priority values to data items to produce a FIFO queue?

In order to produce a queue, give the first item to be inserted a large priority value, and then give each successive item a lower priority. The priorities should be mapped in a monotone decreasing manner, so simply making each priority one less than the previous one should be ideal.

Part B

How would you assign priority values to data items to produce a stack?

To produce a stack, simply give each item a successively higher priority as it is placed in the heap. This time, a monotone increasing priority mapping function is necessary, so simply incrementing each new priority from the last will be ideal.

Laboratory 11: Analysis Exercise 2

Name: Terence Henriod Date: November 14, 2013

Section: 1001

Part A

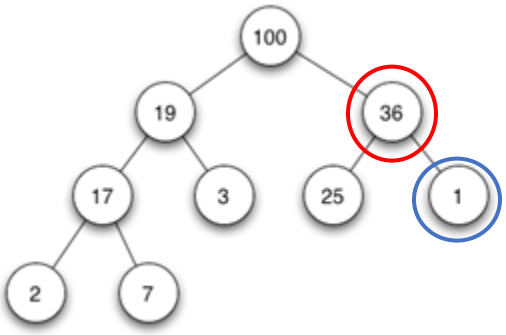
Given a heap containing ten data items with distinct priorities, where in the heap can the data item with the next-to-highest priority be located? Give examples to illustrate your answer.

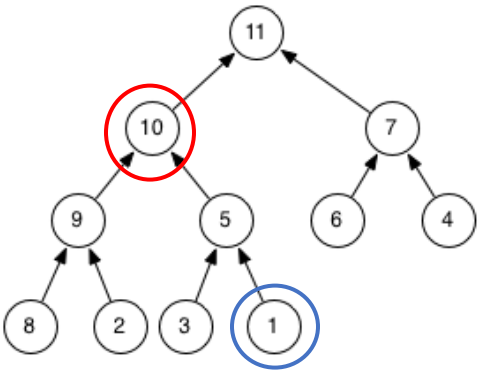
On the second level, assuming this heap places higher priorities in higher levels. No matter what the distinct values are, or which "sub-tree" the next-to-highest item is placed in, the item will percolate upwards until it is only bounded by the highest priority item. (See images below for illustrative examples. Next-to-highest priorities are circled in red.)

Part B

Given the same heap as in Part A, where in the heap can the data item with the lowest priority be located? Give examples to illustrate your answer. .

The item with the lowest priority will always be found in the lowest level of its respective "sub-tree" (again, assuming higher priorities go to higher levels). It is possible that the lowest element will not be found in the absolute lowest level, but if this is the case, the lowest priority item will be found in the next-to-lowest level and in a sub-tree to the right of any sub-tree that contains items in the lowest level. This is because it will be bounded above by any other item, and any item inserted after the lowest priority item will percolate past it during the heap restoration process. (See images below for illustrative examples. Lowest priorities are circled in blue.)





Note: These images do not contain 10 items, but they do still illustrate the properties described above.
Image Sources: http://sir.unl.edu/portal/bios/Binary-Heap.php and
http://scienceblogs.com/goodmath/2008/04/28/binary-heaps/