5SENG001W Algorithms
Week 8
Tutorial Exercises: Stacks, Queues and Heaps

These exercises cover: Stacks, Queues and Heaps.

You will also make use of David Galles' web based *Data Structure Visualizations* tools. During its use it may help to slow the animation right down so that you can follow the steps. You should also take screen shots of your built data structures and/or record its use.

Exercise 1.

(a) Implement a **static** fixed sized (15) stack of integers by completing the given Java class ArrayStack.java, or by defining your own class. The following operations and enquires should be provided:

push(int),   pop(),   clear(),   toString()

top(),   size(),   isEmpty(),   isFull()

(b) Test your implementation by defining test code using suitable test data, that performs all the operations and enquires on your stack. How does your implementation deal with attempting to push onto a full stack or pop from an empty stack?

(c) As part of the testing use David Galles' stack visualisation tool at:

https://www.cs.usfca.edu/~galles/visualization/StackArray.html

Use the same test data as in (b) to visualise your stack by perform the same sequence of test operations on it as in your test program. For the values of the enquiry operations determine by inspection of the stack visualisation. If your test program's output does not match those produced in the tool, make sure you understand what your mistake is and correct your code. You should take screen shots of your testing.

Exercise 2.

(a) Implement a dynamic queue of integers by completing the given Java class ListQueue.java, or by defining your own class. The following operations and enquires should be provided:

queue(int),   dequeue(),   clear(),   length(),   isEmpty(),   toString()

front() - return value of the first item in the queue, but do not remove from queue.

last() - return value of the last item in the queue, but do not remove from queue.

(b) Test your implementation by defining test code using suitable test data, that performs all the operations and enquires on your queue. How does your implementation deal with attempting to dequeue from an empty queue?

(c) As part of the testing use David Galles' visualisation tool for queues at:

https://www.cs.usfca.edu/~galles/visualization/QueueLL.html

Use the same test data as in (b) to visualise your queue by perform the same sequence of test operations on it as in your test program. For the values of the enquiry operations determine by inspection of the queue visualisation. If your test program's output does not match those produced in the tool, make sure you understand what your mistake is and correct your code. You should take screen shots of your testing.

Exercise 3.

(a) Complete the missing steps 6 – 9 from the Heapsort example given in the Week 8 Lecture notes.

(b) Given the input sequence: 6, 3, 7, 4, 2, 8, 1, 5, 0.
Follow the Heapsort steps using *minimal key values* to build the initial *complete* binary tree, then produce the sequence of trees and heaps that until the algorithm is complete.

(c) Compile and run the MinHeap.java program and compare its output with your answer to part (b).

(d) Study the MinHeap.java program and try to understand how it works.

(e) Look at and explore the Heap and Heapsort visualisation tools on David Galles' web site:

https://www.cs.usfca.edu/~galles/visualization/Heap.html

https://www.cs.usfca.edu/~galles/visualization/HeapSort.html

Try to understand how the tools are building the heaps and performing the Heapsort algorithm.