5SENG003W 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 enquiries should be provided:

push(int), top(), pop(), clear(), toString(), 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 enquiry operations, check this by visually inspecting the queue visualisation.

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 enquiries should be provided:

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

(b) Test your implementation by defining test code using suitable test data, that performs all the operations and enquiries 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 performing the same sequence of test operations on it as in your test program. For enquiry operations, check this by visually inspecting the queue visualisation.

Exercise 3.

(a) 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.

(b) Study the MinHeap.java program and try to understand how it works. Compile and run it to see what it does.

(c) The program uses the input sequence: 6, 3, 7, 4, 2, 8, 1, 5, 0 to perform a naïve version of Heap Sort by inserting the values one by one and then extracting them in order.
Add the heapify method as described in the lecture in order to improve this. Then run it again to check your implementation. Try different input sequences as well.