Assignment Specific Code Review Checks

A01 –(https://github.com/anthonyjchriste/ics211f13/wiki/Assignment-1-Sec-3)

- 2 classes – Room, Hotel
- Specified getters and setters
- Interactive driver / guest registration
- Check out - correct guest bill
- Check out – room number, guest name, days stayed, total cost
- Check out – resets room to empty
- Print list of occupied rooms
- Overriden toString method as specified by assignments
- Check in

A01

- Array length = 26
- Initialized with correct characters
- Prompts user (using Scanner.nextInt)
- System.arraycopy use
- Prints out uppercase vowels
- Prints out consonants normally
- Prints out number of non-vowels
- InputMismatchException
- Handles negative/large input

A02

- Multiple constructors
- Distance between points
- Proper formatted toString
- Segment – isHorizontal, isVertical
- Segment - left, right, upper, lower
- Segment – length
- Segment – toString
- Normal segment move
- Horizontal segment move
- CreateSegments class specified as per assignments

A03

- Find on empty array
- Find on front of array
- Find on middle of array
- Find on end of array
- Find missing (should cause MissingResourceException)
- Find duplicates next to each other

- Find duplicates not next to each other
- -1 value for not finding duplicates

A04

- 3 classes – KeyedNode, OrderedLinkedLists, AddressList
- Test size() on – empty list, after add to empty, after add to front/middle/end
- Test add() on – empty, front, middle, end
- Test get() on – empty, front, middle, end
- Test get() with invalid index
- Test find() on – empty, front, middle, end
- Test find() - when item is not in list
- Test that code ignores the case of the characters
- Ensure that items are added into the list at correct position for add methods

A05

- A05 is not an option for code review

A06

- Test simple addition, subtraction, multiplication, division (2 operands and 1 operator, e.g. 1 + 2)
- Simple arithmetic as above, but with floating point values
- Test multiple operator expressions all with low precedence
- Test multiple operator expressions all with high precedence
- Test mixed precedence with low precedence first
- Test mixed precedence with high precedence first
- Test mixed precedence with multiple combinations of low and high precedence

A07

- Test offer null to array/linked queues
- Test offer to empty array/linked queues
- Test offer to non-empty array/linked queues
- Test offer duplicates to array/linked queues at front, middle, and end of queue
- Test that duplicates are not inserted and false is returned
- Test offer on array queue past max capacity (10)
- Test poll on empty array/linked queues (should return null)
- Test poll on single item array/linked queues
- Test poll on >1 item array/linked queues
- Test mixing calls to poll/offer to array/linked queues
- Make sure duplicated are not added

A08

- Test algorithm on example sudoku boards provided by Edo
- This one is particularly challenging for blackbox testing, as there is really only a single method

to test, and it either works or it doesn't.
- Make sure recursion is being used
- Check backtracking algorithm

A09

- toList should return items in-order
- Test toList on empty tree, single item tree, and multi-item tree
- Test toList with left subtree/right subtree
- Test toList on longer left/right subtrees
- Test toString for same items as toList
- All of the above tests also test add(), since lists and strings can't be made before things are added to the tree
- Test ArrayIndexOutOfBounds exceptions for left-array
- Test what happens with larger left-array than needed
- Test what happens when an array of size 0 is passed into addition

A10

- Check Comparator
- Constructor to p-queue takes Comparator
- Priority queue uses heap
- Timing code
- add as defined by Collection, remove as defined by Queue
- Expanding arrays (double size, copy everything from original)
- Add uses iterative method to reheapify
- Remoce uses recursion to reheapify
- Correct implementation of simulation

A11

- Test put() unique key returns null
- Test put() duplicate key returns old value
- Test that keys are associated with values
- Test the chained hashing is being used
- Test null as – first, second, and both parameters to put (should causes NPE)
- Test null as parameter to get (should cause NPE)
- Test getting from empty table (return null)
- Test getting element that DNE in table (return null)
- Test getting a single element
- Test putting and getting multiple elements
- Test that gets after update return updated value
- Test mix of puts/gets