ITCS 2214 Data Structures
Assignment 1

Download the assignment from Moodle. Change the name of the folder to your first-name underscore last-name underscore assign1 (*FirstName_LastName_Assign1*). DO NOT change the structure of the files. If you write your code somewhere else, copy and paste it in the ArrayBaseStack.java and CircularArrayBaseQueue.java. ENSURE you DO NOT paste over the package declaration at the top of those files. Those are the ONLY two files that you should change. DO NOT change the two interfaces. Use Main.java as your driver; put test code in this file to debug your data structures. It is ok if you leave the code in Main.java, I will use my own testing code. In the Read Me file, write a description of each method you coded (what the method is suppose to do). This can get you partial credit just in case your implementation is flawed. Upload a zip folder to Moodle before the deadline.

These are the methods you should complete:

1. Fill in the code in the ArrayBaseStack.java file. The top of the stack should pivot/start at the **upper end** of the array, **NOT** at the lower end that begins at index zero. As data is pushed on the stack, the top should move towards the lower end/indices of the array. **NO other instance or static methods or variable should be used, including a count variable.**

  a. **ArrayBaseStack()**: initialize the stack to the default capacity (1 point)
  b. **ArrayBaseStack(int capacity)**: initialize the stack to the capacity specified by the parameter. (1 point)
  c. **push(T data)**: pushes the data element onto the top of the stack, if the stack is not full. (1 point)
  d. **pop()**: pops the data element from the top of the stack and return it, if the stack is not empty. (1 point)
  e. **bottom()**: returns, but does not remove the data element at the bottom of the stack. (2 point)
  f. **isEmpty()**: returns true if the stack is empty, otherwise returns false. (2 point)
  g. **isFull()**: returns true if the stack is full, otherwise returns false. (2 point)
  h. **size()**: returns the number of data elements on the stack. (2 point)
  g. **toString()**: prints the data elements in the stack from the top to the bottom (one per line). (1 point)

2. Fill in the code in the CircularArrayBaseQueue.java file. You have the freedom to start the queue from any end of the array, I recommend the lower end. **NO other instance or static methods or variables should be used, including a count variable.**

  a. **CirculararrayBaseQueue()**: initialize the queue to the default capacity (1 point)
  b. **CirculararrayBaseQueue(int capacity)**: initialize the queue to the capacity specified by the parameter. (1 point)
  c. **enqueue(T data)**: enqueues the data element at the rear of the queue, if the queue is not full. (2 points)
  d. **dequeue()**: dequeues the data element from the front of the queue, if the queue is not empty, and resets front and rear to their initialized values if the queue is empty after dequeueing. (2 points)
  e. **isFull()**: returns true if the queue is full, otherwise returns false. (2 points)
  f. **isEmpty()**: returns true if the queue is empty, otherwise returns false. (2 points)
  g. **size()**: returns the number of data elements in the queue. (2 point)
  h. **reset()**: shift the data elements in the queue to the end where the queue was initialized from. That is, front is set to the end of the array it started from and rear is set appropriately. If the queue is empty, front and rear are set to their initial values. (4 points)
  i. **reverse()**: reverses the data elements in the queue. Therefore, the data element that was at the front is now at the rear, the second data element is now the second-to-last data element, etc. **Front and rear should NOT be changed after this operation executes.** (4 points)
  j. **shuffle()**: randomly shuffles the data elements in the queue. **Front and rear should NOT be changed after this operation executes.** (4 points)
  k. **toString()**: prints the data elements in the queue, from the front to the rear (one per line). (3 points)