Joseph Martella
Professor Neville
CS 251 Project 2 Analysis
February 11, 2011

- My implementation of the randomized queue involved me using a resizable array. I chose an array because of the randomize feature of the queue. Using the StdRandom class, the shuffle method allowed me to randomize the array easily, and this couldn't be done with a linked list. However, for the deque, I used a doubly-linked list. Simply keeping a pointer to the head and the tail, adding to the front or the back of the deque is simple with a linked list; it is just a matter of pointing these pointers to a new item for adding, or away from an item in regards to deleting.
- Since my randomized queue implementation uses a resizable array, every method has the max array access of N. Most methods actually will never traverse the entire array because it simply shuffles the array and then uses the first index to perform the operation. There will never be more than N array accesses for an operation.
- Each deque operation takes constant time because it is just a matter of switching pointers. No list traversal is necessary for adding or removing items from the deque. The only thing that has to be done is creating the object (for adding) and switching pointers around to incorporate a new item into the array or to delete an item from the array.
- Deque: 3*N*k+ 8 bytes (Node class overhead)
  RandomizedQueue: (kN+16 bytes) + (kN+16 bytes) + 4 bytes (counter) + 2kN bytes (array pointers); k is dependent on what type of Item the queue is holding.