1. How did you test your queue implementations were correct?

I used Test.java to print out the ArrayQueue when the queue resized because it became full. Then I printed it out after dequeueing a few elements and then enqueueing elements until the queue was full to test the ­circular nature of the ArrayQueue after resizing.

1. One option with array stacks and queues is to resize when they become full. Assuming that the initial size is 128, how many times would the array need to resize if you added 1 million items? What about with 1 billion items or 1 trillion items (assuming the computer had enough memory)? Explain how you got your answer.

I know that the log2(128) = 6, and 10 powers of 2 is 1024 or about 1000, so with every ten powers of 2, a number increases by 1000 fold. The log2(1,000,000) is about 20, and 20 – 6 = 14, thelog2(1,000,000,000) is about 30, and 30 – 6 = 24, and the log2(1,000,000,000,000) is about 40, and 40 – 6 is 34, so the array would need to resize about 14, 24, and 34 times.

1. Suppose that instead of using an array or list as the organization structure to implement a queue, you were provided a functional Stack. How would you implement dequeue using one or more stacks? Provide the psuedocode below. (Reminder Stack provides the methods pop, push, peek, and isEmpty)

The dequeue method:

public String dequeue() {

String s = this.queueStack.pop();

this.front = this.queueStack.peek();

this.size--;

return s;

}

1. Supposethatyouhaveenqueuednitems.Thenyoudequeueasingleitem.For that dequeue, what is the time complexity for the list implementation? What about the array implementation? And the Stack implementation? Additionally, what is the space needed for the each of the three implementations?

For the list implementation, the time complexity is O(1) because you save the front item, change front, decrement size, and return the saved item.

For the array implementation, the time complexity is O(1) because you save the front item, increment front, decrement size, and return the saved item.

For the stack implementation, the time complexity is O(1) because you save the popped item, change front to the top of the stack, decrement size, and return the saved item.

The space requirement is O(n) for the number of items enqueued.

1. Include a description of how your project goes “above and beyond” the basic requirements (if it does).

My project goes above and beyond because a resizing component in the enqueue method of ArrayQueue was implemented in addition to the basic requirements. When the queue becomes full, the array is copied over to a new array double its size so that a new item can be enqueued.

1. What did you enjoy about this assignment? What did you not enjoy? What could you have done better?

It was great.

1. What else, if anything, would you like to include related to this homework?

Nothing.