Kelvin Abrokwa-Johnson

Claudia Estes

Write Up

1. a description of any issues your team encountered during the implementation

Implementing the stacks, queues, and deques was overall a pretty simple task. We only encountered problems with our deque class because we forgot to return in many of the methods. That problem was hard to find, but easy to fix.

Our team had a hard time understanding exactly how the Towers of Hanoi work. The base case was difficult to understand, and it took a while for us to figure out what the base case was. The recursive method for the towers is very obscure and it took a long time for us to realize that we could switch around the different stacks. Also, our Linked List class had a bug in it and fixing it set us back in time.

2. a justification for each of your test cases, and an explanation of why those test cases are complete

* Deque
  + push\_front: asserted that when you push front multiple times, the last item pushed front is at the front
  + push\_back: same as push\_front except back
  + peek\_front/peek\_back: peek\_front/peek\_back returns the data you last push\_front/push\_back without removing
  + pop\_front/pop\_back: pop\_front/pop\_back returns the data you last push\_front/push\_back and removes the node from the linked list
  + We also made sure that popping from an empty deque does not throw
* Stack
  + pop: asserted that the stack removed the item from on top of the stack
  + peek: asserted that the top of the Stack was returned, but not removed
  + push: asserted that the stack added an item to the top of the stack
* Queue:
  + enqueue: ensured that the first item enqueued is the first item dequeued
  + dequeue: ensured that the item dequeued is the first item enqueued

3. a brief explanation of how you could use an array as the storage medium instead of a Linked List or Deque for each of the three structures

A circular array would need to be used instead of a Linked List for the Deque, Queue and Stack classes. With the circular array, modular operations could either push an item onto the front or the back of the array for the Deque class, or just the front or back for the Stack and Queue classes. Head and tail pointers would be essential for the array implementation of the classes in order to keep track of the front and back locations for when items begin to loop around the array, or when items are popped off of the front or back.