CENG 112 – DATA STRUCTURES
Homework 3

April 8, 2016

Due Date: April 22nd, 2016

Programming Assignment 1.1 Double-Ended Queues

The queue ADT only allows data to be put at one end and pulled from the other end. A double-ended queue (called deque) allows insertions and removals from both ends.

The deque API is similar to the queue API and has the following four methods instead of enqueue/dequeue:

- `void enqueueFront(Item elem)` inserts a new element at the front.
- `Item dequeueFront()` removes and returns an element from the front.
- `void enqueueBack(Item elem)` inserts a new element at the back.
- `Item dequeueBack()` removes and returns an element from the back.

(a) Implement the Deque API using singly-linked lists in "DequeA.java".

(b) Implement the Deque API using circular doubly-linked lists in "DequeB.java"

Programming Assignment 1.2 More RPN Calculations

In this exercise we will modify the RPN calculator so that it functions the same but the implementation is slightly different in each case.

(a) The code in "RPNCalc.java" uses different methods to traverse and visit the nodes for printing and evaluation. Write a single `traverse` method that takes three `NodeVisitor` objects: A prefix visitor, an infix visitor, and a postfix visitor. It should make calls to each visitor's `visit` method in the correct place if the visitor is not `null`. Reimplement `evalTree` and `printTreeInFix` methods so that they call `traverse` with appropriate `NodeVisitor` objects. Submit your new implementation as "RPNCalcA.java".

Hint: The node visitor for evaluation will need to keep the values of the subtrees visited so far in a private `Stack`.

(b) Modify the code in "RPNCalc.java" so that there are no recursive calls. You are allowed to use as many `Stack` objects as necessary. Submit your new implementation as "RPNCalcB.java".