Jonathan Tsai

Programming Methodology II

Midterm A

Problem 3

1.

**public** **class** ProblemThree {

/\*\*

\* A data structure that can contain int values in sorted order is a binary

\* search tree (BST). In a BST, search takes O(logN) time. However, we must

\* use our data structure from problem 2 (a queue) in our solution.

\*

\* Solution: We can dequeue (remove) values from the queue and then add

\* those values into our BST. Dequeue will take O(1) time, while adding

\* values into the BST will take O(logN) time. Then, we simply implement

\* search for our BST, which takes O(logN) time.

\*

\* Please see "BST.java" for partial BST implementation (ran out of time).

\* Still need to implement search.

\*\*/

/\*

\* Pass in the data structure, Data, from Problem 2. Value is the number to

\* be searched in Data. Return the index if found.

\*/

**public** **int** ProblemThree(Data data, **int** value) {

**return** 0;

}

}

/\*\* Part of Problem 3 \*\*/

**public** **class** BST {

**private** **class** Node {

**private** **int** value;

**private** Node left, right;

// Node constructor

**public** Node(**int** value, Node left, Node right) {

**this**.value = value;

**this**.left = left;

**this**.right = right;

}

}

Node root;

// BST constructor

**public** BST() {

root = **null**;

}

// add value to the BST

**public** **void** add(**int** value) {

root = add(root, value);

}

**public** Node add(Node x, **int** value) {

**if** (x == **null**) {

x = **new** Node(value, **null**, **null**);

}

**if** (value < x.value) {

x.left = add(x.left, value);

} **else** **if** (value > x.value) {

x.right = add(x.right, value);

} **else** {

x.value = value;

}

**return** x;

}

}