Data Structures (CS 102), Professor Yap – Fall 2016

HOMEWORK

November 3, 2016

Homework 5: Due on Thursday Nov 10, 2016

- This homework is worth 100+5 Points. In each homework, you get the extra 5 points by following all our instructions.
- READ CAREFULLY the homework instructions and policy in `http://cs.nyu.edu/~yap/wiki/class/index.php/DataStruc/Hw-page`.

PART A: WRITTEN ASSIGNMENT (50 Points)

Question A.1 (4 Points each part)
Recall the notion of a Collatz sequence **Collatz(n)** for any positive integer n. Please do the following BST Simulation.

(a) Begin with an empty BST, and insert the keys on Collatz(17). Draw the final BST (it should have size 13). You do not have to show intermediate BST's.

(b) Draw the BST after deleting the key at the root. (Be sure to use our convention of "cutting" a successor, not predecessor.)

(c) Here is the pre-order traversal list of **keys** in a BST:

$$(16, 6, 2, 8, 7, 19, 18, 25, 20, 23, 26)$$

Please draw the BST.

(d) Now give the post-order traversal of the BST from previous part.

(e) Suppose we are given the preorder list of the (names of) **nodes** (not keys) in some binary tree, say

$$(c, a, b, h, f, d, e, g, i).$$

We clearly do not have enough information to reconstruct the tree. But suppose we are also given the inorder listing of nodes:

$$(a, b, c, d, e, f, g, h, i).$$

Please reconstruct the binary tree from these 2 lists. HINT: can you reduce to the previous formulation of the problem?

Question A.2 (4 Points)
In String patterns, we can use the meta-character "|" to represent "Boolean OR" operation. However, we do not have a meta-character to represent the "Boolean AND" operation. For instance, suppose (as in midterm) we define a string ss to be a valid password if it satisfies three conditions: (i) no white

space, (ii) has at least one digit, and (iii) has at least one (small or capital) letter. (4) has at least one capital letter. But since patterns has no "Boolean AND", we cannot define a pattern that satisfies all of (i-iii). How do we get around this restriction?

Question A.3 (10 Points)
Assume the following definition of a binary node:

```
class BinNode {
        int key;
        BinNode left, right;
        int size, height, depth;
}
```

Write a method with this header:

```
void fillStats(Node u, int d);
```

If we call `fillStats(u, 0)`, the method will fill in the size, height and depth information for every node in the binary tree $T(u)$ rooted at u.

Question A.4 (12 Points)
We want a data structure called a `Sqew` which has the operations of a **Stack** (pop/push) and that of a **Queue** (enqueue/dequeue):

```
public interface SqewInterface {
    public int pop();
    public void push(int k);
    public int dequeue();
    public void enqueue(int k);
    public boolean isEmpty();
    public boolean isFull();
    public void init();
}
```

IMPORTANT CONVENTION: We have the usual top/bottom of a stack, and front/end of a queue. *We will identify the top of stack with the front of queue.*

Write a Java class to implement this interface as simply as possible, using an `int` array called `sq` of size `MAXSIZE`. The only other variable you need are two `int` variables, called `tos` and `bos` corresponding to top-of-stack and bottom-of-stack. DO NOT USE ANY OTHER VARIABLES. E.g., do not use a `size` variable. Also do not throw any exceptions for empty or full Sqew conditions. You must implement a "circular" Sqew. We assume that `tos` and `bos` are never equal during the operation of this Sqew (we will not bother to enforce this). Morever, these variables have values between 0 and `MAX-1`, inclusive. The empty Sqew condition is when `tos=1+bos`. The next empty slot at the top (i.e. front) of the Sqew is `sq[tos]`. The method `init()` in the interface is used to initialize the values of `tos` and `bos`. NOTE: Your code should be extremely compact (no more than 3 lines in the body of each method).

PART B: PROGRAMMING ASSIGNMENT (50 Points)

All needed files are found in `hw5_Yap.zip` from Piazza/Resources. They must be included in your `src` folder and should be unmodified unless we allow otherwise. Our provided Makefile has three targets `run1`, `run2` and `run3` that must not be modified. The targets `run4` and `run5` are for your own programs below.

Question B.1 Binary Search Tree (25 Points)

We have provided a working BST implementation in `src/bst/BST.java`. Please read the comments in that file to understand our design. There is another file `src/bst/MyBST.java` which extends the `BST` class. In this problem, you need to implement several methods as described in the file:

(a) `int size(Node u);` // number of nodes in the BST $T(u)$.

(b) `int indexOf(int k, Node u);` // number of keys in $T(u)$ that are $\leq k$.

(c) `Node get(int i, Node u);` // the "inverse" of indexOf, i.e., `v = get( indexOf(v.key, u), u)`.

(d) `Node succ(Node u);` // successor of node u

(e) `Node pred(Node u);` // predecessor of node u

(f) `void showTree(Node u);` // gives a sketch of the tree

All these methods should work even if node u is null. Our main method will test your programs with these tests:

`testGetIndex(...)` and `testSuccPred(...)`

Target `run4` has been written for you to run the main program of `MyBST`, with the defaults. Note that `showTree(u)` is useful when we run the program with large values like `nn=1000`, when we cannot afford to display the full tree!

Question B.2 Generic Binary Search Tree (25 Points)

Please write another file `src/bst/BSTz.java` to implement a generic version of `src/bst/BST.java`:

```
public class BSTz <T extends Comparable<?  super T>> {
    private class BinNode <T extends Comparable<?  super T>> {
       T key;
       BinNode<T> left=null, right=null, parent=null;
       //Constructor
       BinNode(T k) { key=k; }
    }//BinNode
          ⋮
}//BSTz
```

Make as few changes as possible in order to achieve your goal, and in particular, do not modify the logic of the methods in `BST`. In your main program, we have the two tests,

`testAdd(...)` and `testDelete(...)`

as before, but the generic class is to be instantiated with `T=Integer`. See p.166,180 of Text for Generic Java classes.

Write your own target `run5` for this program (it should be basically the same as `run4`).