Second Quiz - Version A
CS 1102 Computer Science 2

Spring 2021

Thursday April 1, 2021
Instructor Muller

KEY

Before reading further, please write your name on the top of all of your quiz answer sheets.

This is an open notes and open book quiz. But **collaboration is expressly prohibited**.

- Partial credit will be given so be sure to show your work.
- Feel free to write helper functions if you need them.
- **Please write neatly.**

Problem	Points	Out Of
1		3
2		2
3		5
Total		10

Part 1: Short Answer (3 Points)

All of the problems in Part 1 are half-point problems. Please write your answers in the `answers.txt` file.

1. Java's `==`, `!=`, `<`, `<=`, `>` and `>=` operators can be used with operands of various types. In this and the next question, we're concerned with how they work on integers (i.e., values of type `int`).

   True or false: The `==` operator defines an equivalence relation on integers.

   Answer:

   ```
   True
   ```

2. True or false: The `<` operator defines a total order on integers.

   Answer:

   ```
   False, the < operator isn't reflexive.
   ```

3. We've discussed the idea of *statements* or *assertions* related to sets. For example, the containment statement $2 \in \{1, 3, 5\}$ states (falsely) that 2 is an element of the set $\{1, 3, 5\}$. We've also discussed general statements related to sets using *variables* and their *quantifers* $\forall$ (for all) and $\exists$ (there exists). For example, the statement $\forall n \in \{1, 3, 5\}.n > 4$ can be understood as a conjunction of three statements: $1 > 4$ *and* $3 > 4$ *and* $5 > 4$, another false statement. How can we interpret the statement $\exists n \in \{1, 3, 5\}.n > 4$?

   Answer: $1 > 4$ ***or*** $3 > 4$ ***or*** $5 > 4$**.**

4. Let $A = \{a, b, c\}$. $R = \{(b, b)\}$. Is R a reflexive relation on A?

   Answer:

   ```
   No
   ```

5. Let $A = \{a, b, c\}$. $R = \{(b, b)\}$. Is R a symmetric relation on A?

   Answer:

   ```
   Yes
   ```

6. Let $A = \{a, b, c\}$. $B = \{0, 1\}$. Show two different partial maps from A to B.

   Answer:

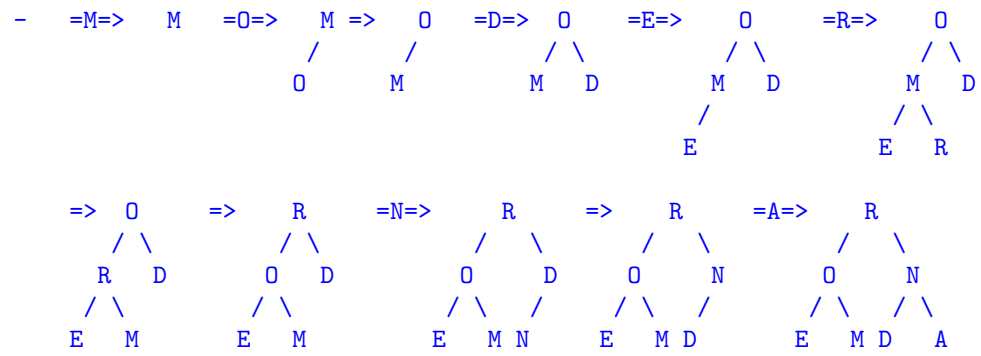
   ```
   {}
   {(a, 0)}
   ```

Part 2: Binary Heaps (2 Points)

Show **all** of the successive complete binary trees that result from the left-to-right insertion of the letters in `MODERNA` into an empty *max binary heap*. I.e., a binary heap in which the root contains the maximum value.

Feel free to draw the sequence of complete binary trees long-hand or to render them electronically. If you draw them long-hand, you'll have to submit an image by git adding and committing it to you local quiz repo and then git pushing to your master quiz repo on GitHub. You can render them electronically in whatever way you want, using simple ASCII-art in the `answers.txt` file or using your favorite drawing tool.

Answer:

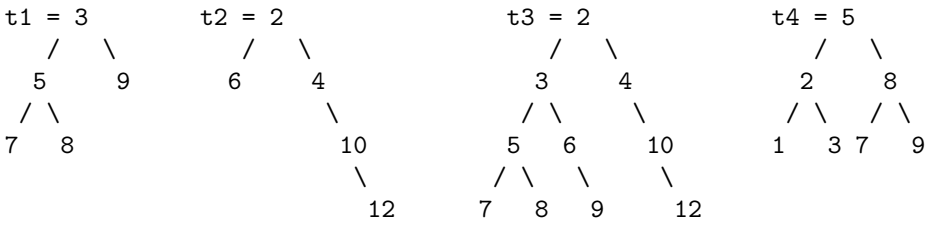


Part 3: Binary Trees (5 Points)

- **There are 3 five-point problems in Part 3. Feel free to work on as many as you would like. If you work on more than one problem, please clearly specify the single problem you've chosen for grading.**
- Feel free to write helper functions.
- Feel free to write your code long-hand or using an editor such as IntelliJ. If you write long-hand, you'll have to submit an image of your code by git adding and committing it to your local quiz repo and then git pushing to your master quiz repo on GitHub.

Consider a binary tree with integers in the nodes:

```
class Node {
  int info;
  Node left;
  Node right;
}
```



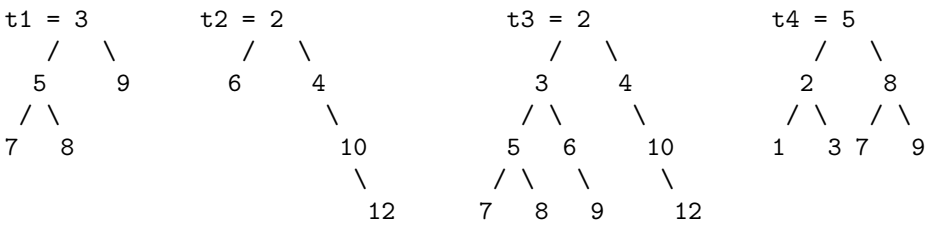
---

1. (5 Points) Recall that a *preorder* walk of a binary tree first visits the root, then walks the left subtree and then walks the right subtree. Write a function `String pathTo(int n, Node a)` such that a call `pathTo(n, a)` returns a string showing the path from `a` to the first item `n` found using a preorder walk of the tree. For example, the call `pathTo(3, t1)` should return the the string `"start"` while the call `pathTo(8, t1)` should return the string `"start left right"`. A call such as `pathTo(80, t1)` should return the string `"no path"`.

   Answer:

```
String pathTo(int item, Node a, String path) {
  if (a == null)
    return "no path";
  else if (a.info == item)
    return path;
  else {
    String leftPath = pathTo(item, a.left, String.format("%s left", path));
    if (!leftPath.equals("no path"))
      return leftPath;
    else
      return pathTo(item, a.right, String.format("%s right", path));
  }
}

String pathTo(int item, Node a) {
  return pathTo(item, a, "start");
}
```



---

2. (5 Points) Let's say that a binary tree is *balanced* if no leaf is at depth greater than $\lfloor \log_2 n \rfloor$ where n is the number of nodes in the tree. For `t1`, n is 5 and $\lfloor \log_2 5 \rfloor = 2$. Since no leaf is at depth > 2, `t1` is balanced. For `t2`, n is also 5 but since 12 is at depth 3, `t2` is unbalanced. Write the function `boolean isBalanced(Node a)`.

   Note: A `log2` function is provided in the harness code. Feel free to use the built-in `Math.floor` function for $\lfloor \cdot \rfloor$.

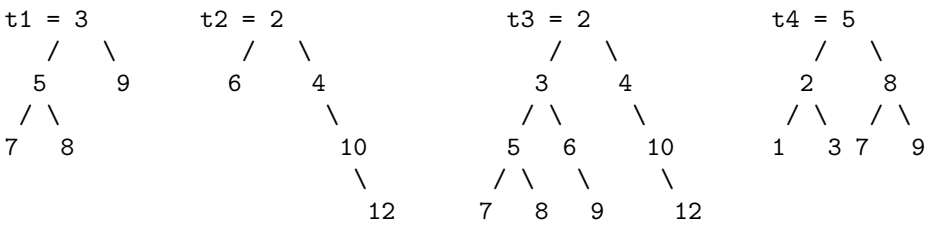
   Answer:

```
int nodes(Node a) {
  if (a == null)
    return 0;
  else
    return 1 + nodes(a.left) + nodes(a.right);
}

int height(Node a) {
  if (a == null || isLeaf(a))
    return 0;
  else
    return 1 + Math.max(height(a.left), height(a.right));
}

static int log2(int N) {
  return (int) (Math.log(N) / Math.log(2));
}

boolean isBalanced(Node a) {
  if (a == null || isLeaf(a))
    return true;
  int n = nodes(a);
  return height(a) <= (int) Math.floor(log2(n));
}
```



---

3. (5 Points) Trees `t1`, `t2` and `t3` are not min binary heaps because they aren't complete binary trees. But they do have the the min binary heap property that the value in every interior node is less than the values at the child nodes. Let's say they're *heapish*. Tree `t3` represents a *merger* of trees `t1` and `t2`. How to merge heapish binary trees `a` and `b`? Well, if either is empty, the other represents the merger. If neither are empty, let's say `b.info` is less than or equal to `a.info`. Then make a new root containing `b.info`. What should the new node's `left` and `right` fields be? Flip a coin. If it comes up *heads*, set the `left` field to `b.left` and set the `right` field to the result of merging `a` with `b.right`. And vice-versa if the coin comes up *tails*.

   Write the function `Node merge(Node a, Node b)` that returns a heapish binary tree whenever `a` and `b` are heapish binary trees.

   Answer:

```
Node merge(Node a, Node b) {
  if (a == null) return b;
  if (b == null) return a;
  if (a.info < b.info) {
    if (StdRandom.uniform(2) == 0)
      return new Node(a.info, a.left, merge(a.right, b));
    else
      return new Node(a.info, merge(a.left, b), a.right);
  }
  else if (StdRandom.uniform(2) == 0)
    return new Node(b.info, b.left, merge(a, b.right));
  else
    return new Node(b.info, merge(a, b.left), b.right);
}
```