AP® COMPUTER SCIENCE AB
2007 SCORING GUIDELINES

Question 3: Tree Ball

Part A: getMaxHelper **4 points**

+1 base case
- **+1/2** test if `current == null`
- **+1/2** return 0

+3 recursive case
- **+1/2** `getMaxHelper` of left subtree
- **+1/2** `getMaxHelper` of right subtree
- **+1 1/2** calculate max
  - **+1/2** `current.getValue()`
  - **+1/2** determine max of left and right subtree max path scores
  - **+1/2** add root value to larger subtree max
- **+1/2** return sum of root value and max score of subtrees

Part B: constructor **5 points**

+2 create & init node
- **+1/2** create new `TreeNode`
- **+1/2** generate random digit 0...9
- **+1/2** store generated value in node
- **+1/2** assign references to left & right

+2 construct tree
- **+1** construct a multi-level tree with both left and right children
- **+1** construct full tree with `numLevel` levels

+1 assign constructed tree to `root`

AP® Computer Science AB
2007 Canonical Solutions

Question 3: Tree Ball

PART A:

```
private int getMaxHelper(TreeNode current)
{
    if (current == null) {
        return 0;
    }
    else {
        int leftMax = getMaxHelper(current.getLeft());
        int rightMax = getMaxHelper(current.getRight());
        if (leftMax >= rightMax) {
            return ((Integer)current.getValue()).intValue() + leftMax;
        }
        else {
            return ((Integer)current.getValue()).intValue() + rightMax;
        }
    }
}
```

PART B:

```
public GameBoard(int numLevels)
{
    root = GameBoardHelper(numLevels);
}

private TreeNode GameBoardHelper(int numLevels)
{
    if  (numLevels == 0) {
        return null;
    }
    else {
        return new TreeNode(new Integer((int)(Math.random()*10)),
                            GameBoardHelper(numLevels-1),
                            GameBoardHelper(numLevels-1));
    }
}
```

AB3-a

(a) Write the `GameBoard` method `getMaxHelper`, which returns the maximum path score that can be obtained from the tree rooted at `current`. A path score is computed by summing the node values along the path from `current` to a leaf node. The maximum path score for an empty tree is 0.

Complete method `getMaxHelper` below.

```
/** @param current the root of the subtree to be processed
 *  @return the maximum path score for the subtree rooted at current
 */
private int getMaxHelper(TreeNode current){
    if (current == null)
        return 0;
    int val = (Integer) (current.getValue()).intValue();
    int lval = getMaxHelper(current.getLeft());
    int rval = getMaxHelper(current.getRight());
    if (lval > rval)
        return val + lval;
    else
        return val + rval;
}
```

Part (b) begins on page 16.

(b) Write the `GameBoard` constructor, which creates a full binary tree with `numLevels` levels. Each node in the tree should contain an independently generated random `Integer` value from 0 to 9, inclusive. Recall that for this question a full binary tree has the property that all leaves are on the same level. You may find it useful to write and use a helper method to create the tree.

Complete the `GameBoard` constructor below.

```
/** Creates a full binary tree rooted at root with numLevels levels
 *  with a random integer from 0 to 9, inclusive, generated for each node
 *  @param numLevels the number of levels in the tree
 *         Precondition: numLevels > 0
 */
public GameBoard(int numLevels){
   root = boardHelper(numLevels);
}
TreeNode boardHelper(int levelsLeft){
   if (levelsLeft == 0)
      return null;
   TreeNode t = new TreeNode(new Integer((int)(Math.random()*
                                          9.9)));  //will truncate from
                                                   0 to 9 with roughly
                                                   equal probability
   t.setLeft(boardHelper(levelsLeft-1));
   t.setRight(boardHelper(levelsLeft-1));
}
```

(a) Write the `GameBoard` method `getMaxHelper`, which returns the maximum path score that can be obtained from the tree rooted at `current`. A path score is computed by summing the node values along the path from `current` to a leaf node. The maximum path score for an empty tree is 0.

Complete method `getMaxHelper` below.

```
/** @param current the root of the subtree to be processed
 *  @return the maximum path score for the subtree rooted at current
 */
private int getMaxHelper(TreeNode current) {
    if (current.getLeft() == null && current.getRight() == null)
        return ((Integer) current.getValue()).intValue();

    if (current.getLeft().getValue() > current.getRight().getValue())
        return ((Integer) current.getValue()).intValue() + getMaxHelper(current.
                                                              getLeft());
    else
        return ((Integer) current.getValue()).intValue() + getMaxHelper(current.
                                                              getRight());
}
```

Part (b) begins on page 16.

(b) Write the `GameBoard` constructor, which creates a full binary tree with `numLevels` levels. Each node in the tree should contain an independently generated random `Integer` value from 0 to 9, inclusive. Recall that for this question a full binary tree has the property that all leaves are on the same level. You may find it useful to write and use a helper method to create the tree.

Complete the `GameBoard` constructor below.

```
/** Creates a full binary tree rooted at root with numLevels levels
 *  with a random integer from 0 to 9, inclusive, generated for each node
 *  @param numLevels the number of levels in the tree
 *         Precondition: numLevels > 0
 */
public GameBoard(int numLevels) {
  Random ran = new Random();
  root = new TreeNode(new Integer(ran.nextInt(9)));

  initialize(root, 1, numLevels, ran);
}
```

```
private void initialize(TreeNode current, int index, int levels, Random ran) {

  index ++;
  if (index < levels) {
     current.setLeft(new TreeNode(new Integer(ran.nextInt(9))));
     initialize(current.getLeft, index, levels, ran);
     current.setRight(new TreeNode(new Integer(ran.nextInt(9))));
     initialize(current.getRight, index, levels, ran);
  }
}
```

GO ON TO THE NEXT PAGE.

AB3-C

(a) Write the `GameBoard` method `getMaxHelper`, which returns the maximum path score that can be obtained from the tree rooted at `current`. A path score is computed by summing the node values along the path from `current` to a leaf node. The maximum path score for an empty tree is 0.

Complete method `getMaxHelper` below.

```
/** @param current the root of the subtree to be processed
 *  @return the maximum path score for the subtree rooted at current
 */
private int getMaxHelper(TreeNode current) {
    if (current == null)
        return 0;
    int max = getMaxHelper(current.getLeft())
            + getMaxHelper(current.getRight());
    return max;
}
```

Part (b) begins on page 16.

(b) Write the `GameBoard` constructor, which creates a full binary tree with `numLevels` levels. Each node in the tree should contain an independently generated random `Integer` value from 0 to 9, inclusive. Recall that for this question a full binary tree has the property that all leaves are on the same level. You may find it useful to write and use a helper method to create the tree.

Complete the `GameBoard` constructor below.

```
/** Creates a full binary tree rooted at root with numLevels levels
 *  with a random integer from 0 to 9, inclusive, generated for each node
 *  @param numLevels the number of levels in the tree
 *         Precondition: numLevels > 0
 */
public GameBoard(int numLevels)
{
    buildTree(root, numLevels);
}

private void buildTree(TreeNode current, int numLev)
{
    if(numLev == 0)
        return;
    Random randNumGen = RandNumGenerator.
                        getInstance();
    current.setValue(randNumGen.nextInt(10));
    buildTree(current.getLeft(), numLev-1);
    buildTree(current.getRight(), numLev-1);
}
```

AP® COMPUTER SCIENCE AB
2007 SCORING COMMENTARY

Question 3

Overview

This question focused on creating and recursively traversing a full binary tree of `TreeNodes`. Students were provided the framework of the `GameBoard` class, which had as a field the root of a tree containing integers at the nodes. In part (a) they were required to implement a recursive helper function, which searches all paths in the tree and determines the maximum path sum. This involved recursively finding the maximum path sum for the left and right subtrees, and then adding the root value to that maximum. In part (b) students were required to implement the constructor for the `GameBoard` class, which could be accomplished using either recursion or iteration.

Sample: AB3a
Score: 8

This response earned all of the points for part (a).

In part (b) the solution earned the ½ point for creating a new `TreeNode`. The generating random digit ½ point was earned. The digits 0..9 are generated even though there is not an even distribution of numbers. The solution correctly stores the generated number in a node and assigns references to left and right nodes.

With respect to the points for tree construction, the helper method creates a multi-level tree and earned 1 point. However, the helper method is missing the return statement (i.e., `return t`), which caused a 1-point deduction of the point awarded for constructing a full tree with `numLevel` levels since the tree is not actually constructed. While the helper does not actually assign the constructed tree to `root`, once the 1-point deduction was made for failure to return the node, the 1 point for assign was awarded since the assignment statement in `Gameboard` properly assigns the tree that would be constructed by the helper to `root`.

Sample: AB3b
Score: 6

This response lost the first two ½ points for failing to guard for an empty tree. The response correctly calls `getMaxHelper()` of the left and right subtrees and earned those two ½ points. The response earned the ½ point for correctly calling `getValue()`. There was a ½ point deduction for not correctly determining the max of the left and right subtrees. Based on the comparison, the solution correctly adds the root value to the larger subtree and returns the sum, earning the last two ½ points.

In part (b) the response correctly creates a new `TreeNode` and earned the first ½ point. The ½ point available for generating a random number was lost because 9 is used instead of 10. This response earned a ½ point each for storing the generated number and assigning references to left and right. This response earned 1 point for creating a multi-level tree and 1 point for assigning the created tree to `root`. The response lost the point for full tree with `numLevel` levels because it has one too few levels.

Question 3 (continued)

Sample: AB3c
Score: 3

In part (a) the response earned the first four ½ points for the work shown that uses a guard check and the two calls of `getMaxHelper()`. There is no call to `getValue()`, no determining of max, and no adding of `root` to the larger subtree, so this solution lost all three ½ points. There is a return but it is not the sum of root value and max score of subtrees, so a ½ point was lost.

In part (b) the response does not create a new `TreeNode`. This omission caused the following deductions: ½ point for create new `TreeNode`, 1 point for construct multi-level, and 1 point for constructing a full tree. Furthermore, because `root` is undefined when it is passed into the helper, it cannot receive a value of any tree constructed by the helper. So the 1 point for assigning constructed tree to `root` was also lost. There is no assignment of left and right subtrees, causing a ½ point deduction for assign references. Finally, this response earned a ½ point for generating a random number and a ½ point for storing the generated number in a node.