Definition: Recursion – see recursion.

Exam covers Chapters 8 and 9, along with the Intellectual Property material.

1. What are the two main components of a well-written recursive method?

2. Trace the execution of a recursive method and determine its output or return value.

3. What happens (precisely) when recursion “runs away” and becomes “infinite”? How is this prevented in a recursive routine?

4. What determines the depth of recursion?

5. For each of the following, write a recursive method to
   a. Print a string backwards
   b. Return the reverse of a string parameter
   c. Print an integer that is the reverse of its integer parameter

6. In what way is recursion related to the concept of a stack?


8. Given the binary tree in Figure 1, show how it would be stored in an array that can be accessed using binary tree protocols (i.e. access methods).

9. Given the tree in Figure 1, what are the leaf nodes? What is the sibling of 4? What are the ancestors of 14? The descendants of 19? What is the depth of the tree?

10. Given the following data, construct a complete binary tree: 43, 21, 17, 5, 12, 4, 27, 53, 1

11. Consider the tree in Figure 2. Is it a binary search tree? Explain your answer. Redraw the tree to be a complete binary search tree.

12. For the binary tree in Figure 2, show the results for each of the following traversals: pre-order, in-order, post-order, breadth-first.

13. What are the basic operations for the BTNode class?
14. Write a method that, given an integer binary tree as a parameter, outputs the contents of just the leaf nodes.

15. Write a method that, given an integer binary tree as a parameter, outputs the contents of just the leaf nodes and their depth. What other parameter(s) do you need?

16. Write a method that, given an integer binary tree as a parameter, returns the sum of only the non-leaf nodes.

17. Write a method that, given an integer binary tree as a parameter, returns the number of nodes in the tree.

18. Write a method that, given a complete binary tree of characters as a parameter, outputs the tree in “pretty print” form, i.e. each child is positioned evenly below its parent. Think in terms of printing the root half way across a line of maximum length n, the root’s left child printing half way between the left edge and the root, etc. An additional parameter would be the maximum length of the print area. Think carefully about the distance between each value being printed out on a line (draw a picture to help). Assume the data in each node is a single character. Can you calculate the maximum depth of a tree that can be printed in this way (how)?

19. For the following expression, draw the corresponding expression tree: A B + C * D E + -

20. For the expression tree in Figure 3, give the expression’s evaluation.

21. For the expression tree in Figure 3, what are the results for a pre-order, in-order, and post-order traversal? Which is the most useful? Why? Are all of these traversals useful? Why or why not?

22. Given the following values (in the order given), draw the resulting binary search tree if insertions into the tree progressed in the same order:
   19, 33, 5, 29, 4, 11, 32, 44, 9

23. Write a method to print out the contents of an integer binary search tree in ascending order.

24. Write a method to print out the contents of an integer binary search tree in descending order.

25. Binary search trees can be a much more efficient representation for searching for stored values versus, say, a linked list representation. Explain. This efficiency is not guaranteed, however. Why not? Explain in detail.

26. Given the following values in any order, draw a binary search tree that demonstrates a worst case tree. A best case tree. 19, 33, 5, 29, 4, 11, 32, 44, 9
27. What are the four main types of intellectual property? Briefly describe each.

28. Explain the principle of “fair use” with respect to copyrights. Give a couple of examples.

29. How do patents, trademarks, copyrights, and trade secrets relate to computing? Give an example of each.
Figure 1:

```
    5
   / \
  4   19
 /     \
21     6
 \
14
```

Figure 2:

```
14
/ \
17 11
/  \/
9  53 4
    /   /
  13 19 19
```
Figure 3:

```
+  
|   *  +  |
|   |   |   |
| 1 | - | 4 |
| 2 | 3 | 5 |
| 6 |
```

Figure 4:

```
45
/    \
/     \  
/      \ 
9       53
/   \
/    \ 
3    17
     /  
   /    
16    47
    /  
   /    
54    
```

Figure 5:

```
<table>
<thead>
<tr>
<th>7</th>
<th>22</th>
<th>29</th>
<th>32</th>
<th>42</th>
<th>50</th>
<th>59</th>
<th>66</th>
<th>69</th>
<th>76</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
```

Figure 6:

```
<table>
<thead>
<tr>
<th>7</th>
<th>22</th>
<th>29</th>
<th>32</th>
<th>42</th>
<th>50</th>
<th>59</th>
<th>66</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
```