Name:_____ Class:_____ Date:_____

Chapter 10: Trees

Reference and copyright:

Author	Title	Publisher	Date	ISBN
Kenneth Lambert	Fundamentals of Python: Data Structures	Cengage Learning; 2nd Edition	October 11, 2018	ISBN-10 : 0357122755 ISBN-13 : 978-0357122754

1. With trees, each item, including the first and last, have a distinct successor.
   a. True
   b. False

2. In a tree, the root item has no parent item.
   a. True
   b. False

3. In a tree, an interior node is a node that has no children.
   a. True
   b. False

4. The height of an empty tree is -1.
   a. True
   b. False

5. A parse tree describes the syntactic structure of a sentence in terms of its component parts.
   a. True
   b. False

6. A file system is similar to a binary search tree.
   a. True
   b. False

7. An access, an insertion, or a removal of a node in vine-like tree with *N* nodes and a height of *N* - 1 would be linear in the worst case.
   a. True
   b. False

Chapter 10: Trees

8. The maximum amount of work that it takes to access a given node in a full binary tree is O(*N*).
   a. True
   b. False

9. The inorder traversal algorithm visits a tree's root node and then traverses the left subtree and the right subtree in a similar manner.
   a. True
   b. False

10. The preorder traversal algorithm traverses the left subtree, visits the root node, and traverses the right subtree.
   a. True
   b. False

11. A min-heap is a binary tree in which each node is less than or equal to both of its children.
   a. True
   b. False

12. The heap sort algorithm builds a heap from a set of data and then repeatedly removes the leaf item and adds it to the end of a list.
   a. True
   b. False

13. An expression tree is never empty.
   a. True
   b. False

14. In the *replace* method for a binary search tree interface, *None* is returned if the first argument cannot be found.
   a. True
   b. False

15. You should use a postorder iteration method for the tree's *__iter__* method to enable the user to create a clone of the tree with the same shape as the original.
   a. True
   b. False

Chapter 10: Trees

16. Two trees are considered equal if they contain the same items in the same positions.
    a. True
    b. False

17. When a linked binary search tree is instantiated, the *self.root* variable is set to *self*.
    a. True
    b. False

18. Because the recursive search algorithm doesn't require a parameter for a tree node, you can define it as a top-level method.
    a. True
    b. False

19. The level order traversal guides the visits to items from left to right through the levels of the tree, much like reading lines of text in a document.
    a. True
    b. False

20. A grammar in a programming language consists of a vocabulary, syntax rules, and a list of operators.
    a. True
    b. False

21. Syntax rules specify how sentences in the language should be interpreted.
    a. True
    b. False

22. One of the types of symbols used by an EBNF is metasymbols.
    a. True
    b. False

23. Array-based binary trees are the easiest to define and implement.
    a. True
    b. False

24. The *peek* method in a heap implementation returns the bottom most item in the heap.

Chapter 10: Trees

a. True
b. False

25. The number of comparisons required for a removal in an array-based heap is at most $\log_2 n$, so the pop operation is O(log*n*).
a. True
b. False

26. In a tree, which of the following is true?
a. all items have a distinct predecessor and successor
b. each item can have multiple children
c. each item can have multiple parents
d. the root has exactly one parent

27. Which of the following is the topmost node in a tree and does not have a parent?
a. root
b. child
c. leaf
d. interior node

28. Which of the following is described as a node's parent, its parent's parent, and so on up to the root?
a. descendant
b. path
c. depth
d. ancestor

29. Which of the following is true about a binary tree?
a. each node has at most two children
b. each node has only one child
c. child nodes can have multiple parents
d. the root node must have only one child

30. What kind of tree would be useful in analyzing the syntax of a sentence?
a. binary search tree
b. sorting tree
c. parse tree

Chapter 10: Trees

d. linear tree

31. Which is true about binary search trees?
a. they cannot support logarithmic insertions
b. they can support logarithmic searches
c. they don't work well for sorted collections
d. each node in the right subtree of a given node is less than that node

32. What is the number of nodes in a full binary tree with a height of 4?
a. 23
b. 19
c. 31
d. 47

33. What is the formula for determining the number of nodes in a full binary tree where *H* is the height of the tree?
a. $2H^{-1} + 1$
b. $2H + 1$
c. $2H^{+1}$
d. $2H^{+1} - 1$

34. What is the height of a full binary tree with 63 nodes?
a. 5
b. 8
c. 6
d. 7

35. Which type of binary tree traversal traverses the left subtree, visits, the root node, and traverses the right subtree?
a. postorder traversal
b. inorder traversal
c. preorder traversal
d. unordered traversal

36. Which type of binary tree traversal visits the tree's root node, the left subtree and then the right subtree?
a. postorder traversal

Chapter 10: Trees

b. inorder traversal
c. preorder traversal
d. unordered traversal

37. Which type of binary tree traversal traverses the left subtree, traverses the right subtree, and then visits the root node?
a. postorder traversal
b. inorder traversal
c. preorder traversal
d. unordered traversal

38. Which of the following is NOT a common application for a binary tree?
a. heap
b. expression tree
c. search tree
d. stack

39. Which of the following is true about a max-heap?
a. each node is less than or equal to its children
b. the largest nodes are nearer to the root
c. the largest items are in the leaves
d. the smallest item is in the root node

40. When the shape of a BST approaches that of a perfectly balanced binary tree, what is the worst case performance characteristic of searches and insertions?
a. O(log$n$)
b. On
c. O(n)
d. O(log$_2n$)

41. What operator causes the __*contains*__ method to run in the binary search tree implementation?
a. =
b. is
c. +
d. in

Chapter 10: Trees

42. What type of traversal occurs in the binary search trees *__iter__* method?

a. sequential

b. postorder

c. preorder

d. inorder

43. In the following code for the *__init__* method for the linked binary search tree, what is the missing code?

```
def __init__(self, sourceCollection = None):
    <missing code>
    AbstractCollection.__init__(sourceCollection)
```

a. self.root = sourceCollection

b. self.root = None

c. sourceCollection.__init__(AbstractCollection)

d. self.leaf = root

44. In the following code for the *find* method, what is the missing code?

```
def find(self, item):
    def recurse(node):
        if node is None:
            return None
        elif item == node.data:
            <missing code>
        elif item < node.data:
            return recurse(node.left)
        else:
            return recurse(node.right)
    return recurse(self.root)
```

a. return node.data

b. return self.data

c. return recurse(node.root)

d. return node.root

45. In the code for the *inorder* method for a binary search tree, what is the missing code?

```
def inorder(self):
    lyst = list()
    def recurse(node):
```

Chapter 10: Trees

```
        if node != None:
            <missing code>
            lyst.append(node.data)
            recurse(node.right)
    recurse(self.root)
    return iter(lyst)
```

a. recurse(node.root)
b. return(node.data)
c. recurse(node.left)
d. return iter(self.root)

46. Which traversal type guides visits to items in the tree from left to right through the levels of the tree?
a. levelorder
b. inorder
c. preorder
d. postorder

47. Which of the following is not a part of a grammar?
a. vocabulary
b. semantic rules
c. syntax rules
d. method rules

48. Which symbol type is NOT found in an EBNF?
a. terminal symbols
b. metasymbols
c. hypersymbols
d. nonterminal symbols

49. Which of the following carries out the actions specified by a sentence?
a. interpreter
b. parser
c. recognizer
d. compiler

50. In the code for the *add* method in the implementation of a heap, what is the missing code?

Chapter 10: Trees

```
def add(self, item):
    self.size += 1
    self.heap.append(item)
    curPos = len(self.heap) - 1
    while curPos > 0:
        parent = (curPos - 1) // 2
        parentItem = self.heap[parent]
        if parentItem <= item:
            <missing code>
        else:
            self.heap[curPos] = self.heap[parent]
            self.heap[parent] = item
            curPos = parent
```

a. curPos += 1

b. break

c. self.heap[curPos] = item

d. parent = curpos