**Question I: Choose the correct answer:**

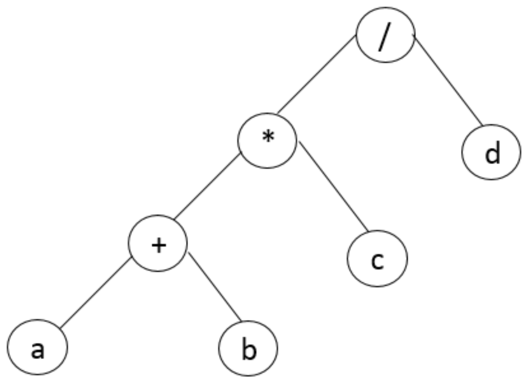

**Figure 1**

1. The postorder traversal of the tree in Figure 1
   a. a b + c d /*
   b. ab+ c * d /
   c. a + b c d / *
   d. None of the above.

2. The preorder traversal of the tree in Figure 1
   a. / * + a b c d
   b. * + a b / c d
   c. a + b c d / *
   d. None of the above.

3. The inorder traversal of the tree in Figure 1
   a. ( ( (a + b) * d ) / c )
   b. / * + a b c d
   c. ( ( (a + b) * c ) / d )
   d. None of the above.

4. The Big O of searching for an item in a binary search tree is:
   a. O( log n)
   b. O ( n )
   c. O ( h)
   d. O ( n/2)

5. On removing an item with two children from a binary search tree, the replacement for the item can be:
   a. The postorder predecessor.
   b. The inorder successor.
   c. a + b
   d. None of the above.

6. How can we test whether a binary tree is a binary search tree?
   a. For each node, if the left child has a key that is less than the node's key, and the right child has a key is greater than the node's key.
   b. Preorder traversal should can return a list ordered in ascending order.
   c. Postorder traversal should can return a list ordered in descending order.
   d. None of the above.

7. Which of the following statements is true about min heaps?
   a. Min heaps are special binary search trees.
   b. Each node's key is greater than all of the node's children's keys.
   c. Min heaps are complete trees.
   d. b + c

8. The most efficient data structure that is used to implement a heap is:
   a. Vectors
   b. Hash tables
   c. Hierarchal binary tree nodes.
   d. None of the above.

9. What's the advantage of heap sort over merge sort?
   a. Heap sort is O ( n ) while merge sort is O ( n log n )
   b. Less usage of memory.
   c. A stable sort.
   d. None of the above.

10. Which of the following sentences is true about Huffman trees?
   a. It can model letters with the same frequency.
   b. Letters with higher frequency tend to have shorter code.
   c. It can be used to decode an English message.
   d. All of the above.

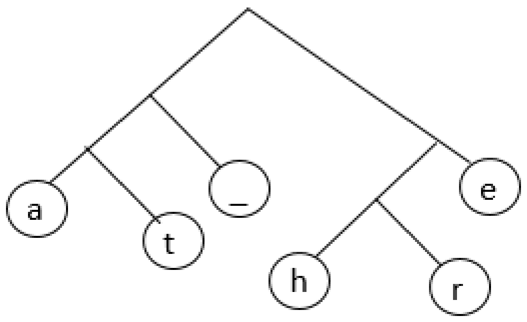

Figure 2

11. Using the Huffman tree in Figure 2, the code of t is:
    a. 111
    b. 000
    c. 010
    d. None of the above.

12. A hash table is a useful data structure in the following situation(s):
    a. We need to determine the element's position in the collection.
    b. We need to find the element fast.
    c. We need to determine the element's relative order in the collection.
    d. All of the above.

13. The most efficient way to resolve collision in a hash table is to:
    a. Linear open addressing.
    b. Quadratic open addressing.
    c. Chaining.
    d. None of the above.

14. On average, the Big O of accessing an element based on its key in a hash-table based map is:
    a. O ( log n)
    b. O ( n )
    c. O (1 )
    d. None of the above.

15. Which of the following statements is true about AVL trees?
    a. Searching for an element is always O ( log n )
    b. An AVL tree can be out of balance by -/+ 1.
    c. An AVL tree is a self-balancing binary search tree.
    d. All of the above.

16. In AVL trees, how many rotations do we need to balance a Right-Right tree?
    a. No rotation is needed.
    b. Two rotations.
    c. One rotation.
    d. Three rotations.

17. Which of the following is true about red-black trees?
    a. The root is black.
    b. A red node can't have red children.
    c. The number of black nodes in any path from the root to a leaf is the same.
    d. All of the above.

18. In the worst case scenario, the height of a an AVL tree is:
    a. 1.44 times the height of a complete binary tree.
    b. $2\ log_2\ n + 2$
    c. $1.002\ log_2\ n$
    d. None of the above.

19. Which of the following is true about 2-3 trees?
    a. A node can have four values.
    b. Leaf nodes have to be at the same depth.
    c. A node can have only up to four children.
    d. All of the above.

20. Which of the following is true about B trees?
    a. They can be used as indexes to large databases stored on disk.
    b. A B tree is a generalization of a 2-3-4 tree.
    c. Nodes can store up to a defined number of items.
    d. All of the above.

**Question II - Understanding Questions**

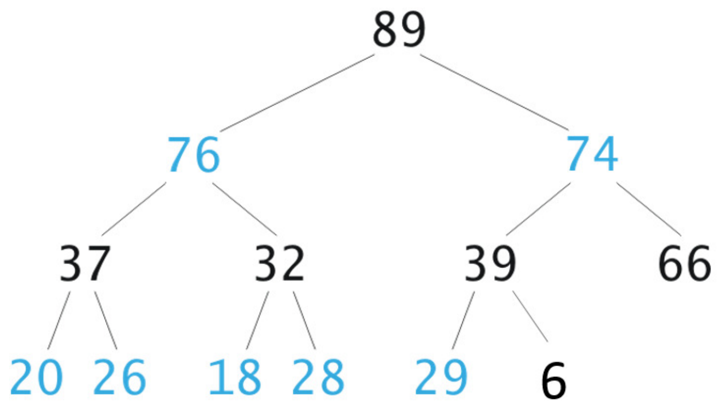

**Figure 3**

1. Show the heap (Figure 3) after removing 89.
2. Build an AVL tree from the following integers: 1, 18, 14, 16, 19, 30.
   Please insert the integers in the specified sequence, and show what happens after inserting every integer.
3. Build the following 2-3-4 tree from the following integers: 5, 18, 14, 16, 19, 20.

**Question III - Programming Question**

Write a **recursive** function that prints the balances of all nodes in a binary tree. (the balance of a node is the height of the right subtree – the height of the left subtree).

**Question IV:**

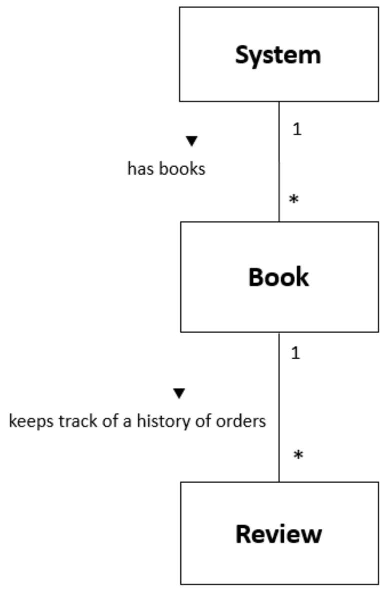

**Figure 3**

A bookstore keeps track of the books it sells. Each book receives reviews from clients that bought the books. The bookstore wants to build a system that keeps track of the books and their reviews. The system should help the staff find a book efficiently by its ID (e.g. ISBN). Furthermore, the system should allow the staff to query the reviews a book received. For instance, it should be fairly efficient to find all the reviews a book has received since the date: May 1, 2000.

**Assumptions:** (1) A book has a name and a title. (2) A review has a rating (a scale from 1 to 10) and a date. Let's assume that the date is unique for each review.

**1. Which abstract data structure would you use to help the system keep track of books? Why?**

**2. Which abstract data structures would you use to help a book keep track of the history of its reviews? Why?**

**3. Modify the minimal class diagram (Figure 3) so that it shows the data structures that you will use to keep track of the objects of the classes (e.g. orders, employees).**

**4. Write pseudo-code algorithms for finding the number of reviews after a given date :** As an example, if the input is January 1, 1998. The algorithm will return the number of reviews for dates greater than January 1, 1998 (e.g. February 20, 1999, March 5, 2000, January 1, 2001, etc.)