CS304 Final Practice Exam
Fall 2016

True/False

Indicate whether the statement is true or false. Computer language questions refer to Java unless otherwise stated.

1. ___**F**___ An object defines the structure of a class.
2. ___**T**___ A class that implements an interface can include methods that are not required by the interface.
3. ___**F**___ The static storage allocation approach creates space for a method when the method is invoked.
4. ___**T**___ Because `QueueUnderflowException` is an unchecked exception, if it is raised and not caught, it is eventually thrown out to the run-time environment.
5. ___**T**___ When comparing two objects using the == operator what is actually compared is the references to the objects.
6. ___**F**___ It is considered OK to add something to a list during an iteration.
7. ___**T**___ A header node does not contain actual list information.
8. ___**T**___ It is possible for a tree node to be both a root and a leaf.
9. ___**F**___ It is easy to access any element of a Priority Queue.
10. ___**F**___ The efficiency of insertion sort is $O(n\log n)$ where n is the size of the list being sorted.

Multiple Choice

11. Which sorting algorithm uses the most amount of space to sort a sequence on *n* elements?
    A. Selection Sort
    B. Insertion Sort
    C. Heap Sort
    D. Quick Sort
    E. Merge Sort

12. How many comparisons would the insertion sort make on an array of 10 elements that is already in ascending order?
    A. 7
    B. 8
    C. 9
    D. 24
    E. None of the above

13. Which of the following statements are **TRUE** about the heap-based implementation of priority queues?
    (1) The `enqueue` method calls `reheapDown`.
    (2) The `enqueue` method calls `reheapUp`.
    (3) The `dequeue` method calls `reheapDown`.
    (4) The `dequeue` method calls `reheapUp`.
    (5) The `enqueue` method adds an element as the root of the heap.
    (6) The `dequeue` method removes the root of the heap.
    A. (1) and (4)
    B. (1), (4), and (6)
    C. (2) and (3)
    D. (2), (3), and (5)
    E. (2), (3), and (6)

14. Which of these represents the **pre-order** traversal of this binary search tree?

```
      15
     /  \
   13    18
  / \    / \
12  14  16  19
```

A. 12, 14, 13, 16, 18, 19, 15
B. 12, 13, 14, 15, 16, 18, 19
C. 15, 13, 18, 12, 14, 16, 19
D. 15, 13, 12, 14, 18, 16, 19
E. 12, 14, 13, 16, 19, 18, 15

15. Which of the following statements is **TRUE** about the merge sort on an array?

A. It divides an array into two subarrays, sorts each half, and merges them back.
B. It divides an array into two subarrays, selects the smallest value from the two subarrays, and repeats the process until the array is sorted.
C. It recursively divides an array into subarrays, merges them, and sorts each subarray.
D. The time complexity of merge sort depends on the initial order of the array.
E. None of the above

16. Which of the following methods is unchanged when moving from a normal linear list implementation to the circular list implementation?

A. `find`
B. `add`
C. `remove`
D. `toString`
E. None of the above

17. What is the value of this postfix expression: `1 2 3 * + 4 2 * 6 5 - - /`?

A. 1
B. 2
C. 3
D. 4
E. None of the above

18. What is returned when invoking the method call `myFun(4)`?

```
int myFun (int n)
{
    if (n <= 1)
        return 1;
    else
        return (myFun(n - 1) + myFun(n / 2));
}
```

A. 3
B. 5
C. 4
D. 10
E. None of the above

19. In a complete binary tree, what is the index number of the parent of the node numbered 20?

A. 8
B. 9
C. 10
D. 11
E. 19

20. What is the time complexity of performing merge sort on an array with *n* elements?

   A. $O(n^2)$
   B. $O(n)$
   C. $O(n\log n)$
   D. $O(\log n)$
   E. None of the above

Short Answers

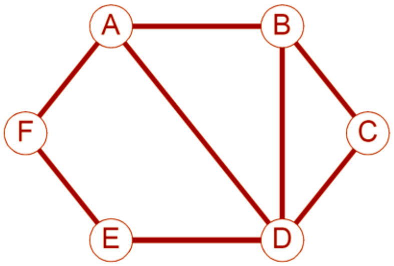
21. What would be the order of the following list after the first two rounds of the Bubble Sort algorithm? You should start from the end.

```
10 2 5 15 20 9 17 8 25 30 4
```

2 4 10 5 8 15 20 9 17 25 30

22. Given a set of vertices and a set of edges for an undirected graph, draw the corresponding graph and list the **Breadth First Search** traversal of the graph, starting from vertex A.

```
V(G) = {A, B, C, D, E, F}
E(G) = {(A, B), (A, D), (A, F), (B, C), (B, D), (C, D), (D, E), (E, F)}
```



BFS traversal: A B D F C E

23. What is the chaining strategy in hashing algorithms? How would it benefit the hashing process?

The chaining strategy in hashing algorithms is to store a reference to a linked list (adjacent list) in each hash table slot. By doing so, elements with same hash values are added to the same linked lists.

The major benefit of using chaining strategy is to avoid collisions which happens frequently in primitive hashing algorithms.

Coding Questions

24. Write a recursive (no "while" or "for" or any looping statement) method "countEvens", which takes an IntNode myLinkedList as the parameter and returns the number of even integers in myLinkedList.

```
public class IntNode
{
    private int m_data;
    private IntNode m_link;

    public IntNode(int data, IntNode link)
    {
        m_data = data;
        m_link = link;
    }
    public int getInfo()
    {
        return m_data;
    }
    public IntNode getLink()
    {
        return m_link;
    }
}

public static int countEvens(IntNode myLinkedList)
{
   IntNode myNode = myLinkedList;

   if (myNode == null)
      return 0;
   else if (myNode.getInfo() % 2 == 0)
      return 1 + countEvens(myNode.getLink());
   else
      return countEvens(myNode.getLink());
}
```

25. Write the method isHeap, which takes an array A of size *n* as the parameter and returns true if the elements in the array between indices 0 and *n* – 1, form a heap, and false otherwise. Running your code should not cause any out-of-bounds exceptions.

```
public static boolean isHeap(int[] A)
{
    int n = A.length;

    int i, left, right;

    for (i = 0; i < n; i++)
    {
        left = 2 * i + 1; right = 2 * i + 2;
        if ((left <= n - 1 && A[left] > A[i])||
            (right <= n - 1 && A[right] > A[i]))
            return false;
    }

    return true;
}
```