1. Given a doublely linked-list, what links need changed and how for:
   (a) Removing to the front of the list
   (b) Adding to the end of the list
   (c) Adding between the 3rd and 4th elements
   (d) Removing the 5th element
2. Know the following terms and concepts:
   (a) Encapsulation
   (b) Polymorphism
   (c) Inheritance
   (d) Abstract
   (e) Interface
   (f) Hashing: Hash function
   (g) Hashing: open addressing
   (h) Hashing: chaining
   (i) Graph: edge
   (j) Three different layout managers
   (k) super
   (l) static
   (m) synchronized
3. Know and be able to identify the following design patterns:
   (a) Model-View-Controller
   (b) Observer
   (c) Decorator
   (d) Iterator
   (e) Command
   (f) Factory
4. Be able to compare the following and provide examples:
   (a) Class vs Object
   (b) i. Checked/Compile-time Exceptions
       ii. Runtime/Unchecked Exceptions
       iii. Errors
   (c) Interface vs Abstract
   (d) HashSet vs TreeSet

5. Understand how to use the following thread functions:
   (a) run()
   (b) join()
   (c) wait()
   (d) start()
   (e) sleep()
   (f) yield()
   (g) notify()
   (h) notifyAll()

6. Indicate which statements are valid:
   (a) ArrayList<String> a = new LinkedList<String> ();
   (b) List<Integer> l = new List<Integer> ();
   (c) Collection<double> c = new ArrayList<double> ();
   (d) Set<String> s = new TreeSet<String> ();
   (e) Collection<Integer,String> m = new HashMap<Integer,String> ();

7. Match each of the first two numbered items with two of the lettered items. All lettered items should be used.

_____1.	Comparable	a)	compareTo(Object o1)
_____2.	Comparator	b)	compare(Object o1, Object o2)
_____3.	———	c)	Generally considered the natural ordering
_____4.	———	d)	Provides an alternative ordering

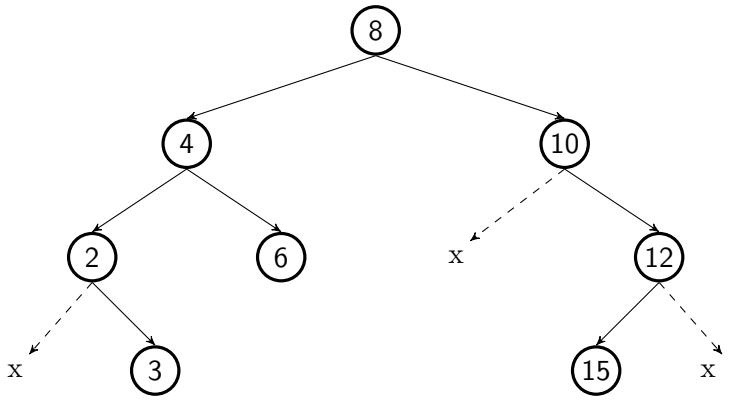
8. This method is defined by Object so every Object has it. Unless overridden by a subclass, it performs the same check as a specific binary operator. What is the method? What is the binary operator? What does String's implementation of it do?

9. Write a method that takes in a boolean. If given true, it should return a new ActionListener that prints "Yes" when triggered. If given false, it should return a new ActionListener that prints "No" when triggered. What two design patterns did you just at least partially implement.

10. Using an interator, remove all even valued numbers from the following list. How can you do it without using an iterator?:

```
1 List<Integer> list = new ArrayList<Integer>(Arrays.asList(6,2,3,4,5,1,7,8));
```

11. Using the tree below, perform the following:



(a) post-order traversal

(b) pre-order traversal

(c) in-order traversal

(d) BFT/BFS

(e) DFT/DFS

12. Build a binary search tree from the following values: 2, 3, 10, 9, 8, 1. Be able to show your steps.

13. Build a min-heap by adding each of the following to the heap in turn: 2, 3, 10, 9, 8, 1. Be able to show your steps.

14. Given a *List<T> list*, fill in the missing code so that the collection is sorted in reverse order. Assume type T is Comparable:

```
1  Collections.sort(list,
2
3
4
5
6
7
8
9
10
11
12 );
```

15. Write a client that sends any message to a server then prints the response.

16. Write a server that accepts a number from a client, creates a thread that sleeps for a random number of seconds between 1 and 10, then responds with "Yes" or "No". The server should be able to handle multiple clients connecting at the same time. After the server sends 10 "No"s, it stops accepting new requests and shutdowns after it finishes handling all existing clients.

17. What will be printed by the following code?

```
1 public class ExceptionTest {
2     public static void process(int num) throws Exception {
3         if(num != 2) {
4             System.out.println("Good number: " + num);
5         }
6
7         throw new Exception("Bad number: " + num);
8     }
9
10    public static void main(String args[]) {
11        try {
12            for(int i = 0; i < 5; i++) {
13                process(i);
14            }
15        } catch(Exception e) {
16            System.err.println(e.getMessage());
17        } finally {
18            System.out.println("Done processing");
19        }
20    }
21 }
```

18. Create a GUI for a digital clock. It displays the time and has two buttons. One button adjusts the time. The second button toggles the alarm. The buttons don't need to function, but the displayed time should grow/shrink as it's window is expanded/condensed. Double bonus points if the time updates every minute and clicking the first button toggles the time between moving forwards or backwards.

19. Indicate which statements are true and which are false:

    (a) (T/F) Integer is a primitive type

    (b) (T/F) String is a primitive type

    (c) (T/F) All classes inherit from Object

    (d) (T/F) Classes can inherit multiple interfaces

    (e) (T/F) Classes can implement multiple abstract classes

    (f) (T/F) We wrap raw I/O streams in buffers to prevent our harddrives from spinning to quickly (thus leading to potential damage and data lost)

    (g) (T/F) PriorityQueues are often implemented by heaps

    (h) (T/F) There are some optimal greedy algorithms

    (i) (T/F) Are all trees are graphs

    (j) (T/F) Are all graphs are trees

    (k) (T/F) Trees may have cycles

    (l) (T/F) To automatically include support for converting an object to/from a sequence of bytes, that object's class needs to implement Savable.

20. All objects put into a TreeSet must implement which interface? Why?

21. What are the best, worse, and average runtime complexities (Big-O) for the following:

(a) QuickSort

(b) MergeSort

(c) HeapSort

(d) Adding an element to a HashMap

(e) Adding an element to the begining of a linked list

(f) Adding an element to the end of a linked list

22. Name two divide-and-conquer algorithms. Explain the differences between them.

23. Convert the following adjacency list into a graph:

```
A: B A
B: A D E
D: B
E: B
F: F
```

24. How does one convert printTree()'s BFT into a DFT? Do it by adjusting the code and updating printTree(). If you're unsure what functions Deques have, make an educated guess.

```
public class Node<T> {
    public T data;
    public Collection<Node<T>> kids;
    public String toString() {
        return data.toString();
    }
}
```

```
public static void printTree(Node<?> root) {
    Node<?> node;
    Deque<Node<?>> frontier = new ArrayDeque<Node<?>>();
    frontier.addFirst(root);
    System.out.print("BFT Tree Output: [");
    while(!frontier.isEmpty()) {
        node = frontier.removeFirst();
        System.out.print("{"+node+"}");
        for(Node<?> kid: node.kids){
            frontier.addLast(kid);
        }
    }
    System.out.println("]");
}
```

25. This review is missing one major algorithm that was convered in class. Which is it?