CSE 143, Winter 2009
Sample Final Exam #1 Key

1.

Statement	Output
`var1.method2();`	`Box 2`
`var2.method2();`	`Jar 2`
`var3.method2();`	`Cup 2/Box 2`
`var4.method2();`	`Jar 2`
`var5.method2();`	`error`
`var6.method2();`	`Pill 2`
`var1.method3();`	`Box 2/Box 3`
`var2.method3();`	`error`
`var3.method3();`	`Cup 2/Box 2/Box 3`
`var4.method3();`	`Jar 2/Box 3`
`((Cup) var1).method1();`	`error`
`((Jar) var2).method1();`	`Jar 1`
`((Cup) var3).method1();`	`Cup 1`
`((Cup) var4).method1();`	`error`
`((Jar) var4).method2();`	`Jar 2`
`((Box) var5).method2();`	`Box 2`
`((Pill) var5).method3();`	`error`
`((Jar) var2).method3();`	`Jar 2/Box 3`
`((Cup) var3).method3();`	`Cup 2/Box 2/Box 3`
`((Cup) var5).method3();`	`error`

2.

```
public class Point3D extends Point implements Comparable<Point3D> {
    private int z;

    public Point3D() {
        this(0, 0, 0);
    }

    public Point3D(int x, int y, int z) {
        // super(x, y);
        // this.z = z;
        setLocation(x, y, z);
    }

    public int getZ() {
        return z;
    }

    public String toString() {
        return "(" + getX() + ", " + getY() + ", " + z + ")";
    }

    public void setLocation(int x, int y) {
        setLocation(x, y, 0);
    }

    public void setLocation(int x, int y, int z) {
        super.setLocation(x, y);
        this.z = z;
    }

    public double distanceFromOrigin() {
        return Math.sqrt(getX() * getX() + getY() * getY() + z * z);
    }

    public int compareTo(Point3D other) {
        if (getX() != other.getX()) {
            return getX() - other.getX();
        } else if (getY() != other.getY()) {
            return getY() - other.getY();
        } else {
            return z - other.z;
        }
    }
}
```

3.

```
public void reorder() {
    if (front != null) {
        ListNode current = front;
        while (current.next != null) {
            if (current.next.data < 0) {
                ListNode temp = current.next;
                current.next = current.next.next;
                temp.next = front;
                front = temp;
            } else {
                current = current.next;
            }
        }
    }
}
```

4.

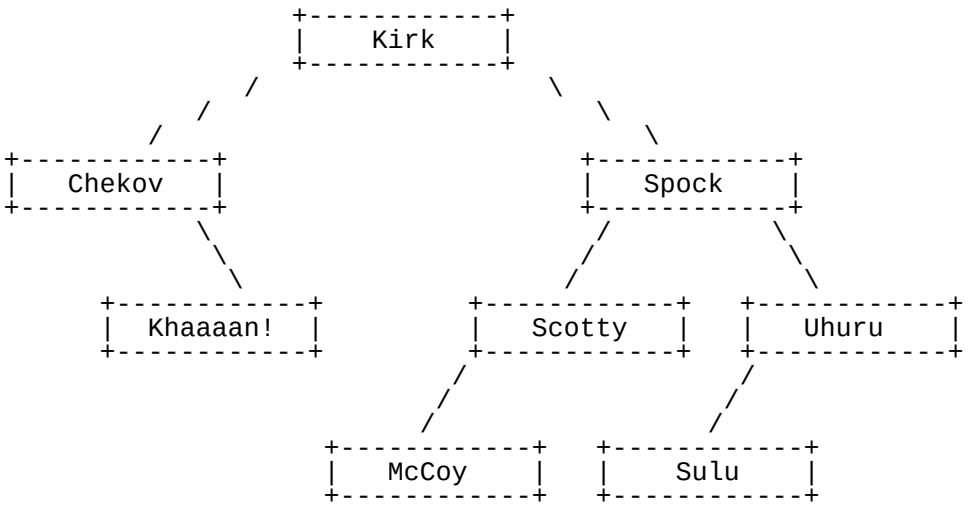
(a) Indexes examined: 7, 3, 5 Value returned: 5

(b) {**12**, 29, 19, 48, 23, 55, 74, 37}
{12, **19**, **29**, 48, 23, 55, 74, 37}
{12, 19, **23**, 48, **29**, 55, 74, 37}

(c)

{37, 29, 19, 48, 23, 55, 74, 12}	
{37, 29, 19, 48} {23, 55, 74, 12}	split
{37, 29} {19, 48} {23, 55} {74, 12}	split
{37} {29} {19} {48} {23} {55} {74} {12}	split
{29, 37} {19, 48} {23, 55} {12, 74}	merge
{19, 29, 37, 48} {12, 23, 55, 74}	merge
{12, 19, 23, 29, 37, 48, 55, 74}	merge

5. (a)



(b)

Pre-order: Kirk, Chekov, Khaaaan!, Spock, Scotty, McCoy, Uhuru, Sulu

In-order: Chekov, Khaaaan!, Kirk, McCoy, Scotty, Spock, Sulu, Uhuru

Post-order: Khaaaan!, Chekov, McCoy, Scotty, Sulu, Uhuru, Spock, Kirk

6.

```
public List<Integer> inOrderList() {
    List<Integer> result = new ArrayList<Integer>();
    inOrderList(overallRoot, result);
    return result;
}

private void inOrderList(IntTreeNode root, List<Integer> result) {
    if (root != null) {
        inorderList(root.left, result);
        result.add(root.data);
        inorderList(root.right, result);
    }
}
```

7.

```
public void construct(int[] data) {
    overallRoot = construct(data, 0, data.length - 1);
}

private IntTreeNode construct(int[] data, int start, int stop) {
    if (start > stop) {
        return null;
    } else {
        int mid = (start + stop + 1) / 2;
        return new IntTreeNode(data[mid], construct(data, start, mid - 1),
                                          construct(data, mid + 1, stop));
    }
}
```