Problem Set 5, Part I

Problem 1: Choosing an appropriate representation

1-1) *ArrayList or LLList?*
Explanation: Since the events will be added in order by date, ArrayList sounds better to use

1-2) *ArrayList or LLList?*
Explanation: Since we need to access the runner’s record frequently and only a few changes are required, ArrayList sounds better to use.

1-3) *ArrayList or LLList?*
Explanation: Since the list should be displayed from the very recent one and the number varies significantly, LLList sounds better to use.

Problem 2: Scaling a list of integers

2-1) For loop have a time complexity of O(n), getItem() have a time complexity of O(n), and addItem() have a time complexity of O(1). So the overall time complexity will be O(n^2).

2-2)

```
public static LLList scale(int factor, ArrayList vals) {
      LLList scaled = new LLList();
      for (int i = vals.length - 1; i >= 0; i--) {
            int val = ((Integer)vals.getItem(i));
            scaled.addItem(val*factor, 0);
      }
}
```

2-3) Yes. Since getItem() hava a time complexity of O(1), the overall time complexity will be O(n).

Problem 3: Working with stacks and queues

3-1)

```
public static void remAllStack(Stack<Object> stack, Object item) {
    Stack<Object> temp = new Stack<Object>();
    while (!stack.isEmpty()) {
        Object top = stack.pop();
        if (!top.equals(item)) {
            temp.add(top);
        }
    }
    while (!temp.isEmpty()) {
        stack.push(temp.pop());
    }
}
```

3-2)

```
public static void remAllQueue(Queue<Object> queue, Object item) {
    Queue<Integer> temp_queue = new LinkedList<Integer>();
    while(!queue.isEmpty()) {
        Object top = stack.remove();
        if(!top.equals(item)) {
            temp.add(top);
        }
    }
    while (!temp.isEmpty()) (
        stack.add(temp.remove());
    }
}
```

Problem 4: Binary tree basics

4-1) 3

4-2) leaf nodes: 4 / interior nodes: 5

4-3) 21 18 7 25 19 27 30 26 35

4-4) 7 19 25 18 26 35 30 27 21

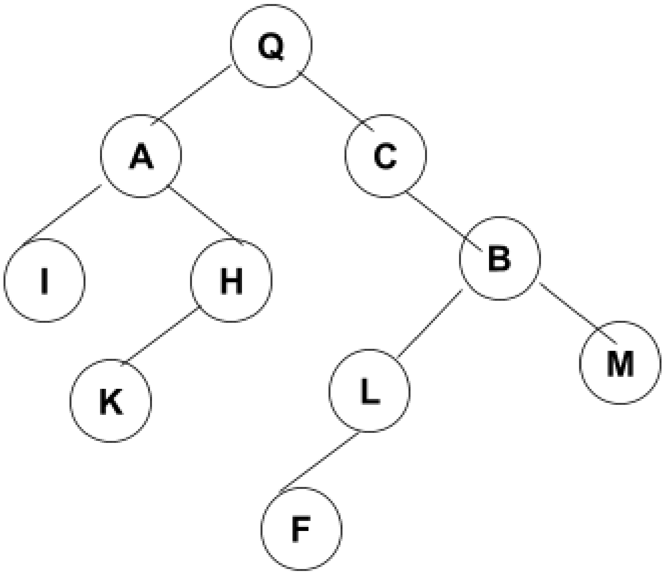
4-5) 21 18 27 7 25 30 19 26 35

4-6) Since each node is greater than every node in its left subtree and each node is less than every node in its right subtree, it is a search tree.

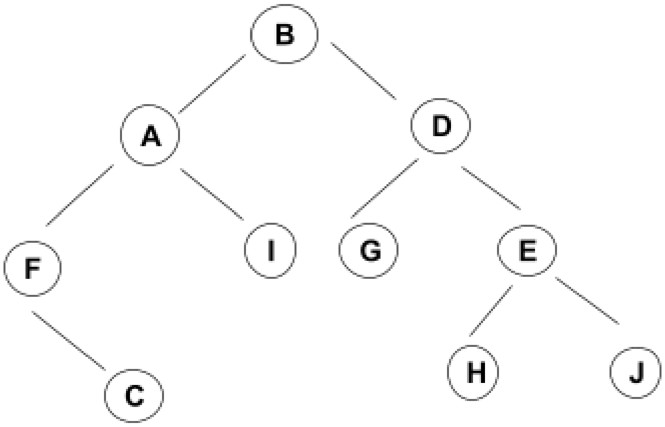
4-7) No. Because the absolute difference between heights of left and right subtrees are different.

Problem 5: Tree traversal puzzles

5-1)



5-2)


B
A
D
F
I
G
E
C
H
J