

[You may use a single side of an 8.5×11 in sheet of paper for reference.]

1. (20 points) True/False (**T** or **F**)

- _____ The `Iterator.hasNext()` method will throw a `NoSuchElementException` called on a completely empty collection.
- _____ The `List.remove()` method is more efficient than the `Iterator.remove()` method since the list knows more about the underlying data structure than the iterator.
- _____ `Iterator` objects throw an `IllegalStateException` if they are asked to retrieve the next element after all elements have been processed.
- _____ When creating JUnit tests, a method annotated with `@BeforeAll` is run once before each method annotated with `@Test`.
- _____ Integration tests should be performed after system tests.
- _____ If `null` is passed to recursive method a base case is not needed.
- _____ You cannot always write an iterative solution to a problem that is solvable by recursion.
- _____ The `Set` interface extends the `Iterable` interface.
- _____ The `Map` interface extends the `Iterable` interface.
- _____ To add a mapping to a `Map`, we use the `put()` method.
- _____ A tree is said to be *full* if all nodes have either two children or no children.
- _____ There is more than one definition for the *height* of a binary tree.
- _____ There is more than one definition for a *perfect* tree.
- _____ There is more than one definition for a *balanced* tree.
- _____ An expression tree is a binary tree.
- _____ A Huffman tree is a binary tree.
- _____ If a class overrides the `hashCode()` method so that it always returns 0, the `equals()` method must be overridden so that it always returns `true`.
- _____ When implementing a hash table using chaining, each element of an array must be `null` or a reference to another data structure.
- _____ When implementing a hash table using open addressing, each element of an array must be `null` or a reference to another data structure.
- _____ When we find the element we are looking for in a hash table, we call this a *collision*.

2. (11 points) Implement the inner class necessary to make the `iterator()` method function correctly.

```
public class ArrayList<E> implements List<E> {
    private Object[] data;
    private int size;

    public ArrayList() {
        size = 0;
        data = new Object[10];
    }

    public E get(int index) {
        if (index < 0 || index > size) {
            throw new IndexOutOfBoundsException("Index: " + index + " Size: " + size);
        }
        return (E) data[index];
    }

    public Iterator<E> iterator() {
        return new ALIterator();
    }

    // ...
}
```

}

3. (8 points) Define the term *adaptor class* and explain how it relates to the `wk8.Queue`.

4. (8 points) Explain the difference between white-box and black-box testing.

5. (8 points) Describe how a binary tree and binary search tree differ.

6. (a) (10 points) Recall the `wk10.BinaryTree<E>` class implemented in lecture with three attributes: `E value`, `BinaryTree<E> left`, and `BinaryTree<E> right`. Implement a `height()` method that returns the height of the tree where the height is zero for an empty tree and one for a tree with one element in it. You may implement a private recursive helper method.

(b) (5 points) Assume the tree is balanced. What is the $O()$ time complexity for your implementation? Justify your answer.

7. (a) (10 points) Recall the `wk10.BST` class implemented in lecture with one attribute `Node<E> root`, and an inner, `Node<E>` class with three attributes: `E value`, `Node<E> left`, and `Node<E> right`. Implement a non-recursive `max()` method that returns the largest element in the tree (or `null` if the tree is empty). You may only implement one method and you may not use any of the methods already implemented in the `BST`.

(b) (5 points) Assume the tree is balanced. What is the $O()$ time complexity for your implementation? Justify your answer.

8. (10 points) Implement the `contains()` method for the `HashTable` class below. Note this is different than the `wk11.HashTable`.

```
public class HashTable<E extends Comparable<? super E>> {  
    private Set<E>[] table;  
  
    public HashTable() {  
        table = new TreeSet[117];  
    }  
  
    }  
  
    }
```

(b) (5 points) If the table always has a length of 117, what is the $O()$ time complexity for your implementation? Justify your answer.