I, the undersigned, do hereby affirm that the work contained in this exam is solely my own, and that none of the results were achieved by cheating. This includes using automated tools to generate answers, stealing the answers off the web, etc. Please do the work yourself.

Name: ________________________________________________________________
(printed legibly)

Signature: ____________________________________________________________

Section: _____________________________________________________________

Student ID __________________________________________________________ (9-digit number)
## Grading Summary

<table>
<thead>
<tr>
<th>Description</th>
<th>Possible</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursion</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Inheritance, Polymorphism</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Lists, Stacks, Queues</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Regular Expressions</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Expression Trees</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Binary Search Trees</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Hashing, Parallel Programming</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Graph: DFS, BFS, MST</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Graph Shortest Path</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Java Coding</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total Points</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
RECURSION

Consider the following program that computes a geometric series using a recursive method:

```java
public class FinalExamRecursion {

    public static void main(String[] args) {
        System.out.println("Converges to " + series(2, 3));
    }

    public static float series(long numer, long denom) {
        // Compute term
        float result = (float) numer / (float) denom;

        if (numer < 0 || denom < 0) {
            // Base case
            return 0.0f;
        } else {
            // Recursive computation
            result += series(numer * 2, denom * 3);
            return result;
        }
    }
}
```

1. (3 points) Describe the geometric series computed by the code above:

2. (2 points) Does the program terminate? Why? After approximately how many recursive calls:
INHERITANCE and POLYMORPHISM

(2 points each) What does the program shown below print? Fill in the blanks:

```java
public class InheritanceProgram {
    public static class A {
        protected int w = 10, x = 20;
        protected int sum() { return w + x; }
    }
    public static class B extends A {
        protected int y = 30;
        protected int sum() { return w + x + y; }
    }
    public static class C extends B {
        protected int w = 0; // Override!
        protected int z = 40;
        protected int sum0() { return w + x + y + z; }
        protected int sum1() { return super.sum() + z; }
    }
    public static void main(String[] args) {
        A a = new A();
        B b = new B();
        C c = new C();
        A poly = new B();
        System.out.println(a.sum()); // Question 3
        System.out.println(b.sum()); // Question 4
        System.out.println(c.sum0()); // Question 5
        System.out.println(c.sum1()); // Question 6
        System.out.println(poly.sum()); // Question 7
    }
}
```

3. _______
4. _______
5. _______
6. _______
7. _______
LIST, STACKS, and QUEUES

(2.5 points each) For problems 6-9, show what the program shown below would print.

```java
public class ArrayListProgram {
    public static void main(String[] args) {
        Character cArray[] = {'A', 'E', '0'};
        ArrayList<Character> list =
            new ArrayList<>(Arrays.asList(cArray));

        list.add(2, 'I');
        list.add(4, 'U');
        System.out.println(list.toString()); // Question 8

        list.remove(3);
        for (int i = 0; i < list.size(); i++)
            list.set(i, Character.toLowerCase(list.get(i)));
        System.out.println(list); // Question 9

        Iterator<Character> it = list.iterator();
        while (it.hasNext())
            if (it.next() < 'e')
                it.remove();
        System.out.println(list); // Question 10

        System.out.println(list.contains('u')); // Question 11
    }
}
```

8. _______________

9. _______________

10. _______________

11. _______________
REGULAR EXPRESSIONS

In regular expressions, [0-9] means any digit, [A-Za-z] means any letter, ? means 0 or 1 occurrences, + means 1 or more occurrences, * means 0 or more occurrences, {2,4} means between 2 and 4 occurrences, {3} means exactly 3 occurrences, . matches any character, and \. matches a period.

12. (7 points) Given the lines of text shown below, put a Yes to the left of lines that are matched and No to the left of lines that are not matched by the regular expression shown below, using a search tool such as the Linux grep command. Remember that grep matches the line only if any contiguous part of it matches the entire regular expression. For full credit, identify which characters in each line are matched by underlining them. The first line is completed as an example:

Regular Expression: [ !@#$%] {2} [0-9] + [A-Z] * .[a-z] {3}

<table>
<thead>
<tr>
<th>Matched?</th>
<th>Line of Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>#@!987654321ABC.txt</td>
</tr>
<tr>
<td></td>
<td>#@987654321ABC.txt</td>
</tr>
<tr>
<td></td>
<td>#987654321ABC.txt</td>
</tr>
<tr>
<td></td>
<td>987654321ABC.txt</td>
</tr>
<tr>
<td></td>
<td>!$A.txt</td>
</tr>
<tr>
<td></td>
<td>!$0A.txt</td>
</tr>
<tr>
<td></td>
<td>!$0AB.txt</td>
</tr>
<tr>
<td></td>
<td>!$0AB..txt</td>
</tr>
<tr>
<td></td>
<td>!@#$0A.jpg</td>
</tr>
<tr>
<td></td>
<td>!@#$0A.jpg</td>
</tr>
<tr>
<td></td>
<td>$%92ABCD-txt</td>
</tr>
<tr>
<td></td>
<td>$%92ABCD.txt</td>
</tr>
</tbody>
</table>

13. (3 points) List three strings that match the regular expression: [ %@&] {3} [0-9] * [A-F] +

1. __________________________
2. __________________________
3. __________________________
EXPRESSION TREES

The expression tree shown below is correct for the following expression:

\[(14 \% 6) + (12 / 2 \ast 3)\]

14. (4 points) Show the postfix expression represented by the tree, with spaces between each token, and no leading or trailing spaces.

15. (4 points) Show the prefix expression represented by the tree, with spaces between each token, and no leading or trailing spaces.

16. (2 points) What does the expression evaluate to, assuming integer math and the normal Java order of operations, which are of course reflected in the prefix and postfix forms and the tree?
BINARy SEARCH TREES

17. *(5 points)* Draw the binary search tree that results from adding the following integers, in exactly the order specified: \([55, 22, 66, 11, 44, 88, 99, 77, 33]\)

18. *(5 points)* Draw the binary search tree from the previous question, assuming that Node 55 has been removed in the manner specified in the Liang slides and textbook:
HASHING and PARALLEL PROGRAMMING

19. (2 points) With linear probing, in which entry in the left hash table will the colliding object with key 15 be stored? Which entry would be used for quadratic probing? The table size is 13, and the hash function is a simple modulo of the integer key. Also list the indices that are tried unsuccessfully.

**LINEAR PROBING:**

**QUADRATIC PROBING:**

<table>
<thead>
<tr>
<th>Index</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>67</td>
</tr>
<tr>
<td>3</td>
<td>81</td>
</tr>
<tr>
<td>4</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>47</td>
</tr>
<tr>
<td>9</td>
<td>61</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>25</td>
</tr>
</tbody>
</table>

20. (4 points) Based on the data structure below, with each hash bucket implemented as a LinkedList, is the search complexity always O(n)? If not, what factor influences the search complexity. This is an essay question, answer on the lines provided:

21. (4 points) Explain the difference between the Runnable interface and the Thread class, with respect to creating multi-threaded programs. This is an essay question, answer below:
WEIGHTED GRAPHS (DFS, BFS, MST)

Note: In all algorithms, the tie breaking rule is to select the node with the lowest value, all other things being equal!

22. (5 points) Please list the order of nodes for a depth-first search (DFS) and breadth-first search (BFS) starting at vertex A. All nodes should be listed:

**DFS:**

**BFS:**

23. (5 points) Create a minimum spanning tree by adding edges in order, according to the algorithm used in the course. Starting at vertex A, the solution has 4 edges. Give the total weight of the tree:

<table>
<thead>
<tr>
<th>Edge</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Total weight: ___
WEIGHTED GRAPHS (SP)

24. (5 points) Find the shortest path between A and all other vertices, according to the algorithm used in the course. At each step, show the set of visited nodes, and the distances and parent arrays:

<table>
<thead>
<tr>
<th>Visited</th>
<th>Distances</th>
<th>Parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>A B C D E</td>
<td>A B C D E</td>
</tr>
<tr>
<td>A</td>
<td>0 ¥ ¥ ¥ ¥</td>
<td>-</td>
</tr>
<tr>
<td>AC</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>ABC</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>ABCD</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>ABCDE</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

25. (5 points) From the arrays shown above, extract the shortest path and its weight between the following vertices:

From A to B) Path: _______ Weight: _____
From A to C) Path: _______ Weight: _____
From A to D) Path: _______ Weight: _____
From A to E) Path: _______ Weight: _____
JAVA CODING

For problems 26-27, write the Java code that is requested.

26. (7.5 points) Write a method that reads a file into an ArrayList<LinkedList<String>>, where each entry in the ArrayList is a LinkedList of the tokens found on that line in the file. Discard empty lines, and split the tokens based on white space. **HINT:** You can use String.split, StringTokenizer, or another Scanner to parse tokens from each line, but the latter is probably the easiest.

```java
public ArrayList<LinkedList<String>> readFile(String filename) {
    ArrayList<LinkedList<String>> list = new ArrayList<>();
    try {
        ...}
  } catch (IOException e) {
    }

    return list;
}
```
27. (7.5 points) Write the method to get an element from a LinkedList, based on the index. The data structure for the LinkedList is the same one used for the assignment on implementing LinkedLists, as shown. Include the bounds check on the index, and throw an IndexOutOfBoundsException if the index is out of range, as specified by the list size variable. Return null if the index is not found. **Hint:** You must traverse the LinkedList to implement the get method.

```java
// Node data structure
class Node {
    public E element;
    public Node next;
}

// Head (first) pointer
private Node listHead;

// Tail (last) pointer
private Node listTail;

// Current size
private int listSize;

// Get method size
public E get(int index) {
    // Bounds check

    // Search list

    // Should never happen
    return null;
}
```