This practice exam has been reviewed by the instructor, but may still contain some errors. In this case we will announce any problems in lecture and we will repost the solution.
DATA STRUCTURES REVIEW

For problems 1-4, show what the program shown below would print (2.5 points each).

**HINT:** Draw a picture of the queue and update the picture as it changes. As a reminder, Queue.offer(e) and Queue.add() inserts to a queue, Queue.remove() and Queue.poll() removes from a queue, and Queue.element() or Queue.peek() reads the queue without modifying it. Queues are first-in first-out (FIFO) data structures.

```java
public static void main(String[] args) {
    Queue<String> queue = new LinkedList<>();
    queue.add("C++");
    queue.add("Java");
    queue.add("C");
    queue.add("Python");
    queue.remove();
    System.out.println(queue.element()); // Question 1
    queue.offer("Java");
    queue.offer("C++");
    queue.remove();
    System.out.println(queue.peek()); // Question 2
    queue.poll();
    System.out.println(queue.peek()); // Question 3
    queue.offer("Fortran");
    queue.offer("C");
    System.out.println(queue); // Question 4
}
```

1. __________________________

2. __________________________

3. __________________________

4. __________________________

REGULAR EXPRESSIONS
Follow the instructions below to write or interpret a regular expression. 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, and parentheses just group items.

5. (2.5 points) Write the regular expression for an account number that starts with the letter 'C', followed by exactly 6 digits from the set '0' to '8' inclusive, followed by a dash '-', followed by 1 or more uppercase letters, and ending with a semicolon ';'.

6. (2.5 points) Write the regular expression for a time string, that starts with the hour (2 digits), followed by a colon ':', followed by the minute (2 digits), optionally followed by a colon ':' and milliseconds (3 digits). The string must always finish with "am" or "pm". The first digit of the hours must be 0 or 1, and the first digit of minutes must be in the range 0..5, and the second digits of hours and minutes are in the range 0..9. For example, 10:59am or 09:15pm, or 04:20:347pm.

7. (2.5 points) List three strings that follow this regular expression: [0-9].[a-zA-Z][2-4]\.bak

_________________________
_________________________
_________________________
8. (4.5 points) Complete the production rules for an assignment statement for a simple language where the variables are groups of one or more letters (uppercase or lowercase), followed by an equals sign '=' followed by a literal integer, which is 1 or more digits, followed by a semicolon ';'. Do not worry about white space. For example, \texttt{xyz = 1234;} or \texttt{onlyLetters = 12345678;}.

\begin{align*}
\text{<assignment>} & ::= \\
\text{<variable>} & ::= \\
\text{<literalInteger>} & ::= \\
\text{<letter>} & ::= \\
\text{<lowerCaseLetter>} & ::= \text{a | b | ... | z} \\
\text{<upperCaseLetter>} & ::= \text{A | B | ... | Z} \\
\text{<digit>} & ::= \text{0 | 1 | ... | 9}
\end{align*}

9. (3 points) Given the following production rules, give three examples of strings that are legal in the grammar defined by the rules.

\begin{align*}
\text{<something>} & ::= \text{A <digit>^* B <punctuation>^?} \\
\text{<digit>} & ::= \text{0 | 1 | ... | 9} \\
\text{<punctuation>} & ::= \% | & | \# | @
\end{align*}

________________________________________

________________________________________

________________________________________
10. (5 points) The table below shows the Huffman codes for each ASCII character in a string with the contents "Tennessee". Encode the string using the table and enter the binary (0 or 1) characters that represent the encoded string below, without any quotation marks. Hint: The encoded string requires 17 binary digits.

<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Character</th>
<th>Frequency</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>T</td>
<td>1</td>
<td>110</td>
</tr>
<tr>
<td>101</td>
<td>e</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>n</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>115</td>
<td>s</td>
<td>2</td>
<td>111</td>
</tr>
</tbody>
</table>

11. (5 points) The tree below shows the Huffman tree that you must use to decode a previously encoded string. Assuming that the encoding is sequence of binary (0 or 1) characters with the value "11101111000011001011010", what is the original string, without any quotation marks.
EXPRESSION TREES

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

$$(12 \times (19 + 1) \mod 12 - 2 \times (12 \mod 5))$$

12. (4 points) Show the postfix expression represented by the tree, with spaces between each token, and no leading or trailing spaces.

13. (4 points) Show the prefix expression represented by the tree, with spaces between each token, and no leading or trailing spaces.

14. (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?
BST MANIPULATION

The BST shown below is about to have some nodes deleted. Please answer questions 15-24 below. (1 point each)

Starting with the BST tree shown above, if we wish to delete node AA:

15. Which node will need to be reconnected (gets a new parent)?
16. Which node will it be reconnected to?
17. Which side of the node will it be reconnected to (LEFT, RIGHT)?
18. Which case of deletion is this, from the textbook (1 easy, 2 hard)?

Starting with the BST tree shown above, if we wish to delete node DD:

19. Which node will need to be replaced?
20. Which node will move to replace it?
21. Which node will need to be reconnected?
22. Which node will it be reconnected to?
23. Which side of the node will it be reconnected to (LEFT, RIGHT)?
24. Which case of deletion is this, from the textbook (1, 2)?

**AVL REBALANCING**

Note: this is more in depth than the knowledge expected on the midterm. If you can do this, the midterm question will be cake.

The BST shown below is unbalanced, and needs your help to become an AVL. Please answer questions 25-34 below. **HINT:** Draw a picture of the rebalanced tree. (1 point each)

![BST Diagram]

25. What is the balance factor for node BB, before rebalancing? __________
26. What is the balance factor for node DD, before rebalancing? __________
27. What is the balance factor for node EE, before rebalancing? __________
28. Is the tree left heavy or right heavy (LEFT, RIGHT)? __________
29. Which rebalancing is required (LL, LR, RR, RL)? __________
30. What node will end up as the root node, after rebalancing? __________
31. What node will end up as the root.left node, after rebalancing? __________
32. What node will end up as the root.left.right node, after rebalancing? __________
33. What is the balance factor for node BB, after rebalancing? __________
34. What is the balance factor for node DD, after rebalancing? __________
Hashing

35. What does a hash function produce? Where is the hashCode method defined in Java? In Java, if you want to use your own custom class as a key in a HashMap or HashSet, what do you need to do? Since a key object that changes hash code after being inserted into a hash map is problematic, it is best to use what sort of objects as keys? (Strings are a good example) Describe the techniques we learned for mitigating hash collisions: linear probing, quadratic probing, and separate chaining.

JAVA CODING

36. (10 points) Write code to insert an element in a BST by descending from the root to where the node should be found. Return false if the element is already in the tree, otherwise insert the element into the correct place and return true. Recursion is not needed. There is an inner class Node has the instance variables element, left, and right, and the element has a compareTo method. HINT: You must keep track of the current node and its parent.

```java
public boolean insert(E e) {
    if (root == null)
        root = new Node(e); // Create a new root
    else {
        // Locate the parent node
        TreeNode<E> parent = null;
        TreeNode<E> current = root;
        while (current != null) {
```

}  
// Create the new node and attach it to the parent node

} 
size++; 
return true; // Element inserted successfully
}