The rules for this exam are as follows:

- **Write your name on the front page of the exam booklet.** Initial each of the remaining pages in the upper-right hand corner. Sign the front of the exam booklet indicating you have followed the KU Academic Integrity Policy. Your exam will not be graded if you have not signed the front page of the booklet.

- This exam will last for 2.5 hours.

- Show **ALL** work for partial/full credit. This includes any definitions, mathematics, figures, etc.

- The exam is closed book and closed notes.

- No laptops, ipads, or other types of non-medical electronic devices are allowed.

- Calculators are allowed provided that they are only used to perform basic computations (and not programmed with algorithms or notes, for example).

- No collaboration of any kind is allowed on the exam.

1. ______ (15 points)  
2. ______ (15 points)  
3. ______ (15 points)  
4. ______ (15 points)  
5. ______ (15 points)  
6. ______ (15 points)  
7. ______ (15 points)  
8. ______ (15 points)  
EC. ______ (15 points)  
T. ______ (120 points)
1. (15 points; 3 points each) Please write **TRUE** or **FALSE** below each question. No additional justification is needed.

(a) The three types of rotations used to balance AVL trees are single, double, and triple rotations.

(b) AVL trees are a generalization of binary search trees.

(c) B-trees are a generalization of binary search trees.

(d) AVL trees are more appropriate than B-trees for use on a disk given their better data locality.

(e) Fibonacci heaps have a better amortized running time than the binary heap and binomial heap.
2. (15 points; 1 point each) Complete the following table on worst case time complexities of the specified operations for the given data structure. For the Fibonacci heap operations, specify the **amortized complexities** instead of the worst case complexities.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Operation</th>
<th>Worst case time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVL tree</td>
<td>insertion</td>
<td></td>
</tr>
<tr>
<td>AVL tree</td>
<td>access</td>
<td></td>
</tr>
<tr>
<td>AVL tree</td>
<td>search</td>
<td></td>
</tr>
<tr>
<td>AVL tree</td>
<td>deletion</td>
<td></td>
</tr>
<tr>
<td>B-tree</td>
<td>insertion</td>
<td></td>
</tr>
<tr>
<td>B-tree</td>
<td>access</td>
<td></td>
</tr>
<tr>
<td>B-tree</td>
<td>search</td>
<td></td>
</tr>
<tr>
<td>B-tree</td>
<td>deletion</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>make heap</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>insertion</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>union</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>find min</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>decrease key</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>deletion</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>delete min</td>
<td></td>
</tr>
</tbody>
</table>
3. (15 points)

(a) (8 points) Insert the following keys, in order, into an empty AVL tree: 21, 26, 30, 9, 4, 14, 28, 18, 15, 10. **Show the resulting AVL tree after each step, and label each rotation.**
(b) (7 points) Now delete 21, 15, and 4 in order. **Show the resulting AVL tree after each step, and label each rotation.**
4. (15 points)

(a) (8 points) Insert the following keys, in order, into an empty B-tree of order 4: 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8. Show the resulting B-tree after each step.
(b) (7 points) Now delete the following keys in order: 8, 5, 12, 7, 2. Show the resulting B-tree after each step.
5. (15 points)

(a) (8 points) Perform a delete min operation on the following Fibonacci heap. **Show the resulting Fibonacci heap after each step.**

![Fibonacci heap diagram](image-url)
(b) (7 points) Decrease 65 to 12 in the following Fibonacci heap. Show the resulting Fibonacci heap after each step.
6. (15 points; 5 points each)
   
   (a) Describe a context in which AVL trees should be used instead of B-trees.

   (b) Describe a context in which B-trees should be used instead of AVL trees.

   (c) Describe a context in which Fibonacci heaps should be used instead of binary heaps.
7. (15 points) Prove that if $n \geq 1$, then for an $n$-key B-tree of height $h$ and minimum order $t \geq 2$,

$$h \leq \log_t \left( \frac{n + 1}{2} \right).$$
8. (15 points) Draw the implementation diagram for the following Fibonacci heap. **Be sure to show all of the pointers.**
OPTIONAL: Extra-Credit Question (15 points)

• (10 points) A red-black tree is a binary search tree in which
  – each node has a color (red or black) associated with it (in addition to its key and left and right children)
  – the following three properties hold:
    1. (root property) The root of the red-black tree is black.
    2. (red property) The children of a red node are black.
    3. (black property) For each node with at least one null child, the number of black nodes on the path from the root to the null child is the same.

Draw an example of a red-black tree which contains at least two red nodes. Be sure to label each node as red or black.

• (5 points) What is your favorite data structure among those covered on the final exam?