This exam has 10 questions, for a total of 100 points.

1. **7 points** Draw the result of inserting a node with key 2 into the following Binary Search Tree.

```
          7
         ↙  ↙
         3   11
        ↙   ↙
        5   9   13
```

**Solution:**

```
          7
         ↙  ↙
         3   11
        ↙   ↙
        2   5   9   13
```

2. **12 points** Which of the following trees are binary search trees? Which of them are AVL trees?

(a)  
```
          12
         ↙  ↙
         4   15
        ↙   ↙
        3   11
```

(b)  
```
          12
         ↙  ↙
         4   1
        ↙   ↙
        3   1
```

(c)  
```
          12
         ↙  ↙
         4   1
        ↙   ↙
        3   1
```

**Solution:**

(a) is a BST and an AVL tree. (4 points)

(b) is not a BST and is not an AVL tree. (4 points)

(c) is a BST but not an AVL tree. (4 points)
3. [10 points] Given the following AVL tree, delete node 10 and re-balance the tree if necessary using one or more left or right rotations. State which rotations you applied to which nodes.

Solution:
To delete node 10, we replace it with the node that comes “after” 10, which is node 11. (5 points) (Could instead replace with 9.)

But now node 8 is not AVL, so we rotate node 8 to the right. (5 points)
4. **12 points** What is the worst-case asymptotic time complexity of rotate \(_k\) in terms of only \(n\) (the length of array \(A\))? The maximum value of parameter \(k\) is \(n\). Provide an argument for why your answer is correct.

```java
static void rotate_1(int[] A) {
    if (A.length > 1) {
        int last = A[A.length - 1];
        for (int i=A.length - 1; i != 0; --i) {
            A[i] = A[i-1];
        }
        A[0] = last;
    }
}
static void rotate_k(int[] A, int k) {
    for (int i = 0; i != k; ++i) {
        rotate_1(A);
    }
}
```

**Solution:** The time complexity of rotate_1 is \(O(n)\) (4 points) and that function is called \(k\) times inside rotate_k (4 points). So the time complexity is \(O(nk)\). Then to express this in terms of only \(n\), the maximum of \(k\) is \(n\), so we have an overall time complexity of \(O(n^2)\). (4 points)

5. **10 points** Apply the Partition algorithm to the following array, ensuring that all elements less or equal to the pivot element are in lower positions and all elements greater than the pivot are in greater positions. The pivot element starts out as the last element of the array. Write down the array after each step (each iteration of the loop), drawing two vertical lines to separate the three partitions (the less-than or equal region, the greater-than region, and the to-do region).

\([8, 9, 0, 6, 7]\)

**Solution:**

\[
\begin{array}{c|c|c|c}
| & 8 & 9 & 0, 6 | 7 \\
\hline
| 8 | 9, 0, 6 | 7 & (2 points) \\
\hline
| 8, 9 | 0, 6 | 7 & (2 points) \\
\hline
| 0 | 9, 8 | 6 | 7 & (2 points) \\
\hline
| 0, 6 | 8, 9 | 7 & (2 points) \\
\hline
| 0, 6 | 7 | 9, 8 & (2 points) \\
\end{array}
\]
The following code implements a doubly-linked list class with a constructor, insert, and erase methods. Fill in the blanks to complete the implementation.

class DNode {
    DNode(int d, DNode n, DNode p) { data = d; next = n; prev = p; }
    int data; DNode next; DNode prev;
}

public class DList {
    DNode sentinel;
    DList() {
        sentinel = new DNode(444, null, null);
        sentinel.next = sentinel; sentinel.prev = sentinel;
    }
    // insert before pos, return iterator pointing to new node
    public Iter insert(Iter pos, int e) {
        DNode b = pos.curr;
        DNode n = new DNode(e, b, ___(a)___);
        b.prev.next = ___(b)___;
        ___(c)___ = n;
        return new Iter(n);
    }
    // erase the node at pos, return iterator to the next element
    public Iter erase(Iter pos) {
        DNode b = pos.curr;
        ___(d)___ = b.next;
        b.next.prev = b.prev;
        return new Iter(___e___);
    }
}

Solution: (2 points each)
(a) b.prev (or pos.curr.prev)
(b) n
(c) b.prev
(d) b.prev.next (or pos.curr.prev.next)
(e) b.next
7. **10 points** What is the best DNA sequence alignment for the sequences ACG and AGT? When computing the score, use +2 for a match, -2 for a mismatch, and -1 for insert and delete. Show your work by filling in the below dynamic programming table and write down the two strings aligned according to the best alignment.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I: -1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D: -1</td>
<td>M: 2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Solution:** Filled in table: (6 points)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I: -1</td>
<td>I: -2</td>
<td>I: -3</td>
</tr>
<tr>
<td>A</td>
<td>D: -1</td>
<td>M: 2</td>
<td>I: 1</td>
</tr>
<tr>
<td>C</td>
<td>D: -2</td>
<td>D: 1</td>
<td>M: 0</td>
</tr>
<tr>
<td>G</td>
<td>D: -3</td>
<td>D: 0</td>
<td>M: 3</td>
</tr>
</tbody>
</table>

The aligned strings: (4 points)

ACG_  
A_GT
8. 11 points Let \( g(n) = \frac{n^2}{10} - 5 \) and \( h(n) = 2n + 3n + 100 \). Give the mathematical definition of Big-O and prove that either \( g(n) \in O(h(n)) \) or \( h(n) \in O(g(n)) \).

**Solution:** The definition of big-O:

\[
 f(n) \in O(g(N)) \text{ if and only if there exists } n_0 \text{ and } c \text{ such that for all } n \\
\text{where } n \geq n_0, f(n) \leq cg(n). \quad (3 \text{ points})
\]

We shall prove that \( h(n) \in O(g(n)) \). (2 points)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( g(n) )</th>
<th>( h(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.9</td>
<td>105</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>150</td>
</tr>
<tr>
<td>100</td>
<td>995</td>
<td>600</td>
</tr>
<tr>
<td>200</td>
<td>3,995</td>
<td>1,100</td>
</tr>
</tbody>
</table>

We choose \( n_0 = 100 \) (2 points) and \( c = 1 \) (2 points). Then we need to show that for every \( n \) that is greater or equal to 100, \( 2n + 3n + 100 \leq \frac{n^2}{10} - 5 \). For \( n = 100 \) we have \( 2n + 3n + 100 = 600 \) and \( \frac{n^2}{10} - 5 = 995 \). (2 points) Each time \( n \) grows by 1, \( h(n) \) grows by 5 and \( g(n) \) grows by \( \frac{2n + 1}{10} \), which is greater than 5 when \( n \geq 100 \). (2 points)

9. 6 points Write down the sequence of keys from the following binary search tree, ordered according to the post-order traversal strategy.

```
    4
   / \
  2   5
 /   / \
1   3   6
```

**Solution:** 1,3,2,6,5,4
10. [12 points] Implement in Java the `max_element` function whose signature (input and output types) is specified below. The function should return the greatest integer that is an element of the input sequence of non-negative integers. If the sequence is empty, the function should return -1 to signify that there was no greatest element.

```java
interface Sequence {
    Iterator begin();
    Iterator end();
}

interface Iterator {
    int get();
    Iterator advance();
    boolean equals(Iterator other);
    Iterator clone();
}

class List implements Sequence {
    ... } // class definition

class Main {
    public static void main(String[] args) {
        int A[] = {3, 1, 4, 2};
        List C = new List(A);
        assert max_element(C) == 4;
    }
}

static int max_element(Sequence S) {
    static int max_element(Sequence S) {
        int m = -1; // (2 points)
        for (Iterator i = S.begin(); // (2 points)
            ! i.equals(S.end()); // (2 points)
            i.advance()) {
            if (i.get() > m) { // (2 points)
                m = i.get(); // (2 points)
            }
        }
        return m;
    }
}
```