April 25, 2007 Name _____

EECS 311 Data Structures
Midterm Exam
Don't Panic!

1. (10 pts) In the boxes below, show the **AVL** trees that result from the successive addition of the given values. Show the nodes, links and balance factors. Draw intermediate trees and clearly indicate rotations, if any, and in what direction.

1. After adding 35 to an empty tree.	2. After adding 87 to the previous tree.
3. After adding 64 to the previous tree.	4. After adding 78 to the previous tree.
5. After adding 81 to the previous tree.	6. After adding 85 to the previous tree.

2. (10 pts) In the boxes below, show the **red-black** trees that result from the successive additions of the given values. Use doubled lines for red links Draw intermediate trees and clearly indicate recolorings and rotations, if any, and in what direction.

1. After adding 35 to an empty tree.	2. After adding 87 to the previous tree.
3. After adding 64 to the previous tree.	4. After adding 78 to the previous tree.
5. After adding 81 to the previous tree.	6. After adding 85 to the previous tree.

3. (10 pts) Draw the **B-trees** that result when adding the following values in succession, starting with an empty tree. Assume each node can only hold 2 keys. To save drawing time, you can choose to draw a new tree only when a split occurs, but make it clear which value caused the split.

Values: 35, 87, 64, 78, 81, 85, 22, 31

4. (5 pts) Give the Big-Oh complexity *with a reasoned argument* for the following algorithm (in pseudo C++) for finding the position in s1 of a longest common substring of two strings s1 and s2, of lengths M and N, respectively. `string::compare()` returns 0 for equality, like C's `strcmp()`.

```
for i from 0 to M
  for len from 1 to M - i
    for j from 0 to N - len
      if s1.compare(i, len, s2, j, len) == 0
         if len > result_len
           result = i
           result_len = len
```

5. (10 pts total) a) Assume a 10-element hashtable, with hash(x) = x mod 10 and linear probing. Show what locations would be probed, in order, for each value in the table, and put the value in its final resting place, if any, in the array:

Value	Locations probed
4371	
1323	
6173	
4199	
4344	
9679	
1989	

Array:

0	1	2	3	4	5	6	7	8	9

b) Repeat, with the same hash(), but using double hashing with hash2(x) = 7 – (x mod 7).

4371 mod 7 = 3	1323 mod 7 = 0	6173 mod 7 = 6	4199 mod 7 = 6
4344 mod 7 = 4	9679 mod 7 = 5	1989 mod 7 = 1	

Value	Locations probed
4371	
1323	
6173	
4199	
4344	
9679	
1989	

Array:

0	1	2	3	4	5	6	7	8	9

6. (10 pts) Using the (space-wasting) C++ tree and node classes below, implement `rotateRight()` so that `node.rotateRight()` rotates `node` clockwise (rightward) through its parent. Each node has a pointer to its parent and a flag indicating if it's a right child of the parent. Drawing a picture first is not required but strongly recommended. Be sure to update all affected fields of all affected nodes.

```
template <typename T> class Tree {
  private:
    struct Node {
      Node *parent, *left, *right;
      bool isRight;
      T data;
      void rotateRight();
    …};
    Node * root;
…};
```

```
template <typename T> void Tree::Node::rotateRight()
{




}
```