Test 3 Criteria

Problem 1 – Code Tracing

1) “uvipZv” ___ helPMe ___
2) “VnQrnkneQ” ___ ImTRappED ___
3) “vmPmvgfs” ___ inSMithS ___
4) “kvkcbwcr” ___ xenCodeR ___
5) “evcapGmrL” ___ refAcToRY ___

Scoring: Point Value: +6/ea
Incorrect letter: -1/ea
Min: +0/ea
[e.g. For (1) they get 1 point for each correct letter.]
[For (2), they start off with 6 points and lose a point for each incorrect.]
[No individual part can have a negative score.]

Total: 30pts
Min: 5pts if they tried

Problem 2 – Coding

```
TreeNode *successor(TreeNode *start)
{
  TreeNode *ret = start;
  if (ret == NULL) return NULL;                    // NULL chk: +3
  if (start->right == NULL)                        // Leaf chk: +3
  { // Leaf node (i.e. no right child => no child successor)
    // Properly handles: (out of 5)
    //   Parent is successor:   +2
    //   Ancestor is successor: +2
    //   No successor:          +1
    while (!ret->isLeftChild && ret->parent != NULL)
      ret = ret->parent;
    ret = ret->parent; // this will be NULL if it has no successor
  }
  else
  { // Non-leaf (i.e. has right child => has child successor)
    // Properly handles: (of 6)
    //   Has right subtree:    +3
    //   Has no right subtree: +3
    ret = ret->right;
    while (ret != NULL && ret->left != NULL)
      ret = ret->left;
  }
  return ret; // Return result: +3
}

// Scoring:  Part A: 20pts
// Note on "Leaf chk.": For the purposes of grading, this is
//   also correct if they checked both the left and right pointer
//   to be NULL, as this is the nature of the problem description.
```

```
TreeNode *find(TreeNode *base, double data)
{
  double cmp;
  if (base == NULL) return NULL;                  // NULL chk: +1
  cmp = data - base->data;                // Valid comparison: +1
  if (cmp == 0) return base;                          // Match: +2
  else if (cmp < 0) return find(base->left, data);  // Left:  +3
  else if (cmp > 0) return find(base->right, data); // Right: +3
  return NULL; // not necessary, an above case will always be taken
}

// Scoring:  Part B: 10pts

TreeNode nodeAfter(TreeNode *root, double data)
{
  // Calls successor(): +4  (-2/ea for incorrect argument)
  // Calls find():      +4  (-2/ea for incorrect argument)
  // Correctness:       +2
  return successor(find(root, data));
}

// Scoring:  Part C: 10pts
```

Scoring: Total: 40pts
Min: 5pts if they tried all 3 parts

Problem 3 – Code Errors

A) OK
B) OK
C) C
D) OK
E) OK
F) OK
G) OK
H) R
I) OK
J) C

K) OK
L) C
M) OK (L)
N) OK
O) C (causes compile errors later)
P) C
Q) OK
R) C
S) OK
T) C

Scoring: Correct answers: +3/ea
Incorrect answers: -1/ea
Total: (max) 30pts