Exam 1

Name:

UMBC email: **@umbc.edu**

DO NOT OPEN YOUR EXAM UNTIL INSTRUCTED!

DO NOT TRUST YOUR GRADE TO A STAPLE: PUT YOUR NAME ON EACH PAGE

Write clearly! Illegible answers will receive no credit.

True or False (2 points each, 20 points total)

Circle the word TRUE or FALSE. If you have not clearly circled one or the other, you will be given no points.

1. TRUE or FALSE
   In a balanced binary tree, the number of nodes grows exponentially with the height.

2. TRUE or FALSE
   Segmentation faults *only* occur when a program dereferences a null pointer.

3. TRUE or FALSE
   The function $3n^2 + 5n^2 \log n + n$ is O(n^2).

4. TRUE or FALSE
   Inserting a new node into a balanced binary search tree is O(n).

5. TRUE or FALSE
   When a function call is executed, the function parameters and local variables are stored on the process's stack.

6. TRUE or FALSE
   If $T(n)$ is $O(f(n))$ then there is a constant $c > 0$ such that $T(n) \leq cf(n)$ for all non-negative n.

7. TRUE or FALSE
   In a double-ended queue, both `insertBack()` and `insertFront()` are $O(1)$.

8. TRUE or FALSE
   Postorder traversal can be used to evaluate arithmetic expressions.

9. TRUE or FALSE
   Binary search can be used to efficiently search a linked list.

10. TRUE or FALSE
    The *height* of a tree and the *maximum depth* of the nodes in the tree are equal.

11. Consider the following partial implementation of a “fast” Range Average Query. Assume that `_table` is defined as a `float *` and `_size` is an `int`. (15 points)

```
// FastRAQ Constructor
FastRAQ::FastRAQ( vector<float> data ) {
   _size = data.size();
   _table = new float[_size];
   _table[0] = data[0];
   for (int i = 1; i < _size; i++) {
      _table[i] = _table[i-1] + _data[i];
   }
}
// Query function
float FastRAQ::query( int i, int j ) {
   if (i == 0) {
      return _table[j] / (j + 1);
   } else {
      return ( _table[j] - _table[i-1] ) / (j - i + 1);
   }
}
```

(a) Describe, mathematically, what is stored in index `j` of `_table`.

(b) Determine asymptotic bounds on the running time of the FastRAQ constructor.

(c) Determine asymptotic bounds on the running time of the query function.

12. Construct an AVL tree from a given set of insertions and deletions (10 points).

(a) Draw the AVL tree that results from the following insertions (read from left to right): 15, 9, 24, 18, 21. Clearly label the nodes (*a*, *b*, c*)* and (*x*, *y*, *z*) in the trinode restructure.

(b) Continuing with the tree from part (a), draw the AVL tree *after* removing the root node (value 15). Describe the steps you used to produce the new tree; clearly label the *w* and *y* nodes and explain how you found *y*.

Asymptotic Analysis (15 points)

13. Give the asymptotic bounds. State your answer in terms of *n.* Justify your answers.

Upper bound: O( )

Lower bound: Omega( )

```
for (i = 0; i < n; i++)
  for (j = i; j < n; j++)
    A[i][j] = A[j][i] = 0;
```

Justification:

14. Give the asymptotic bounds. State your answer in terms of *n.* Justify your answers.

Upper bound: O( )

Lower bound: Omega( )

```
In a binary search tree,
locating the inorder successor
of a node w (the left-most node
of w's right subtree).
```

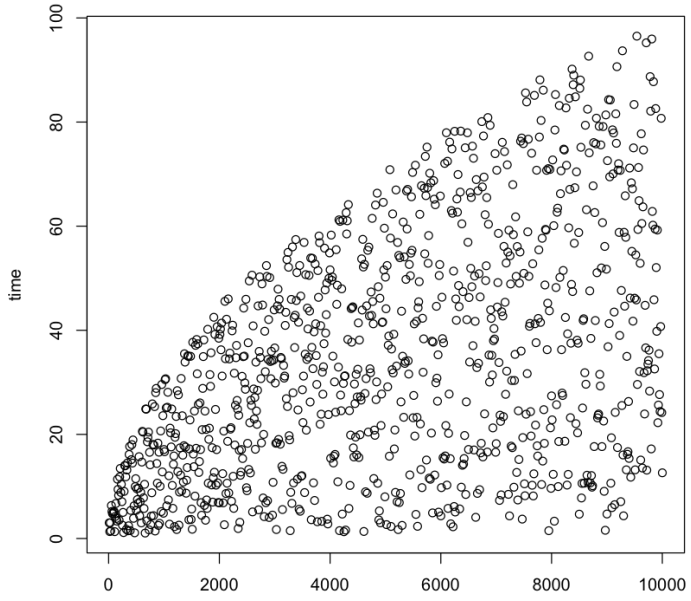
Justification:

15. Suppose $T(n)$ is the running time of an algorithm and a is a positive constant. Use the definition of Big-Oh to show that if $T(n)$ is $O(f(n))$ for some positive, increasing function $f(n)$, then $T(n) + a$ is also $O(f(n))$.

16. You are supervising an intern who has been given the task of writing code to perform the `query()` function in a Block Decomposition implementation of Range Average Query. (10 points)

(a) What are upper and lower asymptotic bounds on the running time of `query()` with a Block RAQ?

(b) The student comes to you at the end of the day and says they have completed the assignment. In addition, they created a graph, shown below, of the running time on their workstation of a number of random queries for a range of values of *n*. The intern is concerned that something is wrong with their code. What do you say?



Coding (30 points)

Another reminder to write clearly. Code should be C++-ish. It does not need to be syntactically perfect, but it should express the correct algorithm.

Consider a doubly-linked, circular linked list class `CircularList`. Each node of the list has next and previous node pointers as well as an integer data field (see below). The `CircularList` class has a private class variable, `Node *_ptr`, that points to a node in the list. We wish to write a function `countValue()` that counts the number of occurrences of a particular value in the list. For example, if the list contains three nodes with the value seven, then the function call `countValue(_ptr, 7)` should return the value three. Also, the function should throw an `invalid_argument` exception if the list is empty. Write an implementation of the function `countValue()`.

Partial implementation of the Node class:

```
class Node {
public:
  Node *_next;
  Node *_prev;
  int _data;

  Node(int data): _data(data), _next(nullptr), _prev(nullptr) {
    // Empty constructor body
  }
};
```

Declared in the CircularList class:

```
int CircularList::countValue( Node *start, int value ) {
  //
  // Your code here
  //
}
```

Space for coding is on the next page!

[Space for coding here]

Extra Credit: Array-based Binary Trees (5 points)

Describe an array-based implementation of a binary tree. Is your implementation time efficient? Space efficient?