CSC 212 Practice Midterm Exam 2B
Problems marked with (*) are challenging and problems marked with (**) are hard

Your Name: ______________________________

1. (10 points) Consider the following linked list declaration. Assume pointers are 4 bytes, integers are 4 bytes and characters are 1 byte. Give a formula for the number of bytes used by a linked list of size n.

```
class LinkedList {
    struct Node {
        Node* next;
        char data[20];
        // ...
    }

    // ...

    Node* m_head;
    Node* m_tail;
    size_t m_size;

    // ...
}
```

2. (10 points) Implement the `max_multiplicity` functions. Assume `m_head` is the head of a **sorted** linked list containing zero or more elements. The **multiplicity** of an element is the number of times it occurs. Return the **maximum** multiplicity. Your implementation must run in $\mathcal{O}(n)$ time.

```
class Multiset {
    struct Node {
        Node* next;
        int data;
        // ...
    };

    // ...

    Node* m_head;

    // ...

    static size_t max_multiplicity(const Node* head) {
        // TODO: Implement this function.
    }

public:
    size_t max_multiplicity() const {
        // TODO: Implement this function.
    }
};
```

3. (10 points) (*) Implement the `first_one` function **recursively**. Assume `v` contains zero or more 0's followed by one or more 1's. Return the index of the first 1. Your implementation must run in $\mathcal{O}(\lg n)$ time.

```
size_t first_one(const std::vector<int>& v) {
    // TODO: Implement this function.
}
```

4. (10 points) What does `L(6)` return?

```
int L(int n) {
    if (n == 0) return 2;
    if (n == 1) return 1;
    return L(n - 1) + L(n - 2);
}
```

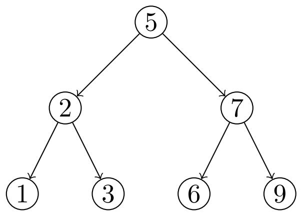
5. (10 points) Give a recurrence relation and base case for $T(n)$, the time complexity of the `mergesort` implementation provided below. Assume `merge` runs in $\Theta(n)$ time.

```
void merge(std::vector<int>& v, size_t left, size_t mid, size_t right) {
    // ...
}

void mergesort(std::vector<int>& v, size_t left, size_t right) {
    if (right <= left + 1) return;
    size_t mid = left + 2 * (right - left) / 3;
    mergesort(v, left, mid);
    mergesort(v, mid, right);
    merge(v, left, mid, right);
}

void mergesort(std::vector<int>& v) {
    return mergesort(v, 0, v.size());
}
```

6. (10 points) Give a formula $C(n)$ for the number of complete 3-ary trees on n nodes.

7. (10 points) (*) Find a closed form for $T(n) = 3T(n/2) + 1$ where $n \geq 2$ and $T(1) = 1$. Assume n is a power of 2.

8. (10 points) Delete $5, 7, 1$ from the following binary search tree. Use the **successor** replacement strategy. Draw the resulting tree after each deletion.



9. (10 points) Insert $4, 8, 1, 6, 0, 3, 9$ into an initially empty 2-3-4 tree. Draw the resulting tree after each insertion.

10. (10 points) Explain the correspondence between red-black trees and 2-3-4 trees. Draw a diagram to illustrate your explanation.