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

Your Name: ______________________________

1. (10 points) Implement the `size` functions. Assume `m_head` is the head of a **sorted** linked list containing zero or more elements. Return the number of **unique** elements in the list. Your implementation must run in $\mathcal{O}(n)$ time.

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

    // ...

    Node* m_head;

    // ...

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

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

2. (10 points) Consider the following queue declaration. Assume the member functions are implemented as efficiently as possible using only the declared member variables. Give a Θ-bounds on the time complexity of `push`, `pop`, `pop` and `size`.

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

    // ...

    Node* m_head;

    // ...

public:
    // ...

    void push(int data);
    void pop();

    int front() const;
    size_t size() const;
}
```

3. (10 points) Implement the `is_sorted` function **recursively**. Assume `v` contains zero or more elements. Return `true` if and only if `v` is sorted in non-decreasing order. Your implementation must run in $\mathcal{O}(n)$ time.

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

4. (10 points) Draw the recursion tree generated when calling `T(5)`.

```cpp
int T(int n) {
    if (n == 1 || n == 2) return 1;
    if (n == 3) return 2;
    return T(n - 1) + T(n - 2) + T(n - 3);
}
```

5. (10 points) Find a closed form for $T(n) = 2T(n-1) + 1$ where $T(0) = 1$ and $n \geq 0$.

)

6. (10 points) Give a recurrence relation and base case for $L(n)$, the number of leaves in **full** binary tree with n nodes. Assume that $n \geq 1$ and n is odd.

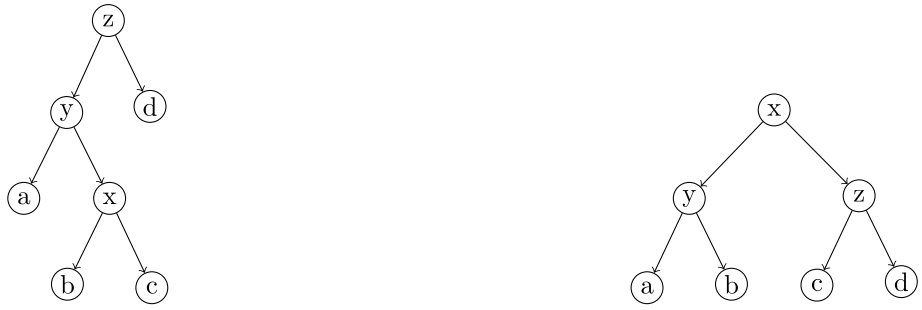
7. (10 points) (*) Implement the mergesort functions. Assume v contains zero or more elements. The merge function takes two sorted subarrays, v[left:mid] (the subarray starting at v[left] up to but **not** including v[mid]) and v[mid:right] (the subarray starting at v[mid] up to but **not** including v[right]), and merges them into a single sorted subarray v[left:right] in $\Theta(n)$ time. Your implementation must run in $\mathcal{O}(n \lg n)$ 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) {
    // TODO: Implement this function.
}

void mergesort(std::vector<int>& v) {
    // TODO: Implement this function.
}
```

8. (10 points) (*) An **LR-rotation** is the following transformation:



(a) Binary tree before LR-rotation at z.

(b) Binary tree after LR-rotation at z.

Implement the `lr_rotate` function. Assume that in the subtree rooted by `root`, `z`, `y`, and `x` are not `nullptr`. Return the new root after rotation. Your implementation must run in $\mathcal{O}(1)$ time.

```
struct Node {
    Node* left;
    Node* right;
    // ...
};

Node* lr_rotate(Node* root) {
    // TODO: Implement this function.
}
```

9. (10 points) Insert $4, 7, 1, 9, 0, 6, 3$ into an initially empty B-tree with $m = 3$. Draw the resulting tree after each insertion.

10. (10 points) Insert $2, 8, 5, 0, 7, 1, 4$ into an initially empty red-black tree. Draw the resulting tree, including colors, after each insertion.