Midterm Review Sheet: COMP15 Fall 2020

About the exam

- On-line via GradeScope for all of 28–29 October (00:01 on the 28^{th} through 23:59 on the 29^{th} Medford time.
- Covers all material through binary search trees (Lab 6). The exam will not cover AVL trees, though one problem below mentions them.

About this review sheet

- **This is not a practice exam.**
- It *does* give you a chance to practice and apply your knowledge

Practice Problems

1. Time Complexity

$T(n) = n^2 + \frac{3}{2} n \log n + 3$	

`for (int i = 0; i < n; i++) {`
`  for (int j = 0; j < 7; j++) {`
`  // constant work`
`  }`
`}`	

`for (int i = 0; i < n; i++) {`
`  for (int j = 0; j < n; j++) {`
`  // constant work`
`  }`
`}`	

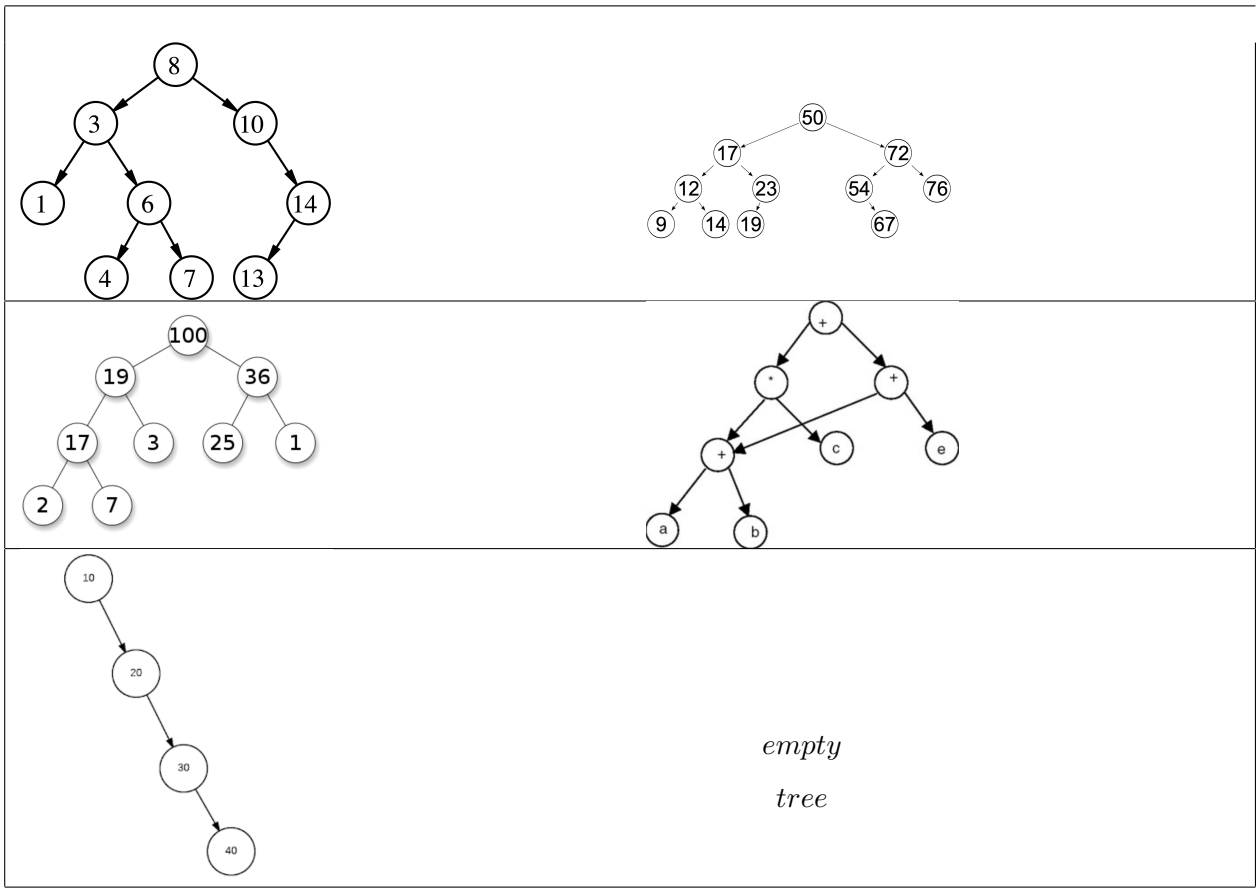
`for (int i = 0; i < n * n; i++) {`
`  for (int j = 0; j < n; j++) {`
`  // constant work`
`  }`
`}`	

`for (int i = 0; i < n * n; i++) {`
`  for (int j = 0; j < 5n - 3; j++) {`
`  // constant work`
`  }`
`}`	

`for (int i = 0; i < 2 * n; i++) {`
`  for (int j = 0; j < i; j++) {`
`  // constant work`
`  }`
`}`	

2. To Tree or Not To Tree

   Given the following linked structures, determine whether it is a **Tree, Binary Tree, Binary Search Tree, or AVL Tree**. Be as **specific** as possible; if a structure is both a BST and a Binary Tree the answer would be BST. If, the structure is none of the above, then say **Not a Tree**.



3. Miscellaneous

   (a) For each of **Unsorted Array, Sorted Array, Unsorted Linked List, Sorted Linked List**, say whether binary search can be performed. Justify your answer.

4. Zip

Write a function that *zips* two linked lists. The zip operation creates a list by inserting nodes from two other lists in alternating positions.

$A = 5 \rightarrow 7 \rightarrow 17$

$B = 12 \rightarrow 10 \rightarrow 2 \rightarrow 4 \rightarrow 6$

$Zip(A, B) = 5 \rightarrow 12 \rightarrow 7 \rightarrow 10 \rightarrow 17 \rightarrow 2 \rightarrow 4 \rightarrow 6$

Note that the extra nodes of the longest list were appended to the end

```
struct Node {
        DataType data;
        Node    *next;
};
/**
 * @brief      "Zips" the two lists beginning by a and b
 *
 * @param      a     The front of list a
 * @param      b     The front of list b
 *
 * @return     The pointer to the head of the zipped list
 *
 * @note       This operation will be done *inplace*.
 *             That is, there should be no dynamic memory
 *             allocation (new) or memory deallocation (delete)
 */
Node *zip(Node *a, Node *b) {









}
```

5. Stacks and Queues.

   Implement a Queue using 2 Stacks.

```
struct Queue {
    stack<int> s1, s2;
    void enqueue(int i);
    int  dequeue();
};

void Queue::enqueue(int i) {









}

int Queue::dequeue() {









}
```

6. Something something something

   Your boss needs an ArrayList that holds **positive** numbers. But they get confused really easily, so they ask you to reduce the number of member variables. That is instead of the usual `size`, `capacity` `ints` and a pointer to `data`, your ArrayList should *only* have a pointer to data and a pointer to "something."

```
class ArrayList {
public:
        void insert(int i);
private:
        ...
        int *data;
        int *something;
};
```

   (a) Describe how you would design this ArrayList for your boss. Be sure to mention what each pointer points to.

   (b) Describe how you would insert an element into your ArrayList, and how it would expand.