This homework covers elementary data structures (arrays and linked lists), binary search trees, and AVL trees. It is due in class Friday, February 14.

Write your solutions carefully — your work should be neat, readable, and organized.

For the "design a data structure" problems (ADM 3-4 and 3-8), use the target runtimes stated in the problem to steer you towards a solution — keep in mind what you know about the running time of standard ADT, array, and linked list operations as well as the tactics discussed in class for improving runtimes (e.g. store instead of search). Your writeup should identify what is stored and how, outline how each operation is carried out, and explain how the desired running time is achieved. Key here is an appropriate level of abstraction — the goal is to express the algorithm with sufficient detail to be understood and to be able to assess its correctness and running time, but not to overwhelm the ideas with details that obscure understanding. Review (and revise) your writeup with an eye towards achieving the right level of detail.

See the Policies page on the course website for information about revise-and-resubmit, late work, and academic integrity as it applies to homework.

1. ADM 3-7, page 103. Write a body for the `remove` method below which runs in $\Theta(1)$ time using the strategy referenced in the problem and discussed in class. Be sure to handle any special cases, such as removing the first element. Use the `ListNode` class given in figure 1.

```
/**
 * Remove the element todel.getValue() from the list.
 *
 * @param head head of the list
 * @param todel node containing the element to remove
 * @return the head of the list after the removal
 */
public ListNode remove ( ListNode head, ListNode todel ) { ... }
```

2. ADM 3-4, page 103.

3. ADM 3-8, page 103.

```
public class ListNode {
  public ListNode ( int value ) { ... }
  public ListNode ( int value, ListNode next ) { ... }

  public int getValue () { ... }
  public void setValue ( int value ) { ... }

  public ListNode getNext() { ... }
  public void setNext ( ListNode next ) { ... }
}
```

Figure 1: `ListNode` class for #1.

---

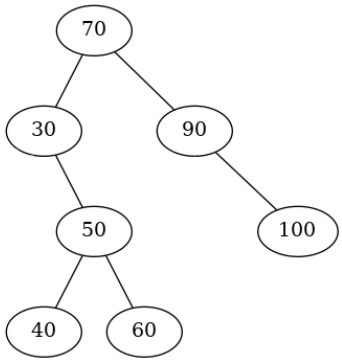
4. Apply each of the operations below to the BST shown, drawing the tree resulting from each operation. (Treat them as separate operations, not a sequence — apply each to the BST shown, not the result of the previous step.) If swaps are needed, swap the element with its successor.

   (a) insert 45

   (b) insert 85

   (c) insert 5

   (d) insert 50

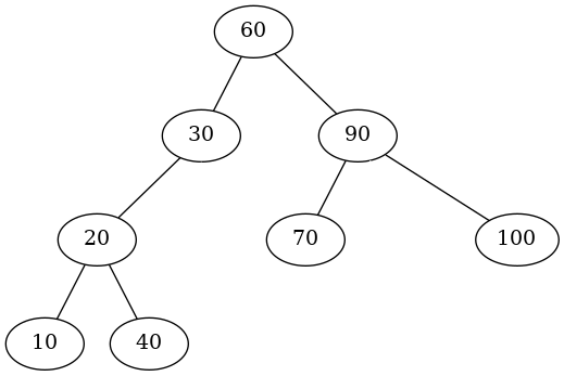


5. Apply each of the operations below to the BST shown, drawing the tree resulting from each operation. (Treat them as separate operations, not a sequence — apply each to the BST shown, not the result of the previous step.) If swaps are needed, swap the element with its successor.

   (a) remove 40

   (b) remove 60

   (c) remove 30



6. Insert the elements 6, 25, 73, 18, 1, 19, 40 into an initially-empty AVL tree, drawing the tree after each insertion. (Treat the operations as a sequence — insert each element into the tree resulting from the previous step.) If swaps are needed, swap the element with its successor.

7. [corrected!] Remove the elements 12, 5 (not 4), 40, 93 from the AVL tree shown, drawing the tree after each operation. (Treat the operations as a sequence — remove each element from the tree resulting from the previous step.) If swaps are needed, swap the element with its successor.