# Data Structures
# CSCI H343, Fall 2022 [B]

**Midterm**

**Name:** _____

This exam has 11 questions, for a total of 100 points.

1. [12 points] What is the output of running the `main` method of class `C`?

```
class Point {
    int x, y;
    Point(int x_, int y_) { x = x_; y = y_; }
}
class C {
    static int add(int x, int y) { x += y; return x; }
    static Point add(Point p, Point q) {
        p.x += q.x; p.y += q.y; return p;
    }
    static void println(int n) { System.out.println(n); }
    public static void main(String[] args) {
        int x = 4; int y = 3;
        int z = C.add(x,y);
        x = x + 1; y = y + 1;
        println(x); println(y); println(z);

        Point p = new Point(x,y); Point q = new Point(x,y);
        Point r = C.add(p, q);
        p.x = p.x + 1; q.x = q.x + 1;
        println(p.x); println(q.x); println(r.x);
    }
}
```

**Solution:** (2 point each)

```
5
4
7
11
6
11
```

2. [10 points] Fill in the blanks to complete the following implementation of the Binary Search algorithm on an array of Boolean values. Given an array that is sorted (false before true) in the half-open range `[begin,end)`, the function `find_first_true_sorted` should return the index of the first true element in the range `[begin,end)`, or if there is none, it should return `end`.

```
int find_first_true_sorted(boolean[] A, int begin, int end) {
    if (___(a)___) {
        return end;
    } else {
        int n = end - begin;
        int middle = ___(b)___;
        if (___(c)___) {
            return find_first_true_sorted(A, begin, ___(d)___);
        } else {
            return find_first_true_sorted(A, ___(e)___, end);
        }
    }
}
```

**Solution:** (2 points each)

```
a) begin == end
b) begin + (n / 2)
c) A[middle]
d) middle
e) middle + 1
```

3. [10 points] Fill in the blanks to complete the following implementation of the `Sequence` and `Iter` interfaces using a linked list.

```
interface Sequence<T> {
    Iter<T> begin();
    Iter<T> end();
}
```

```
interface Iter<T> {
    T get();
    void advance();
    Iter<T> clone();
    boolean equals(Iter<T> other);
}
```

```
class LinkedList<T> implements ___(a)___ {
   Node<T> head;
   public class ListIter implements Iter<T> {
      Node<T> position;
      ListIter(Node<T> pos) { position = pos; }
      T get() { return position.data; }
      void advance() { ___(b)___ }
      Iter<T> clone() { return ___(c)___; }
      boolean equals(Iter<T> other) {
        return ___(d)___;
      }
   }
```

```
    Iter<T> begin() { return ___(e)___; }
    Iter<T> end() { return new ListIter(null); }
}
```

**Solution:** Rubric: 2 points each

a) `Sequence<T>`
b) `position = position.next;`
c) `new ListIter(position)`
d) `position == (ListIter)other.position`
e) `new ListIter(head)`

4. [10 points] Implement the `find` method for the following `BinarySearchTree` class.

```
class BinarySearchTree<K> implements OrderedSet<K> {
    Node<K> root;
    int numNodes;
    BiPredicate<K, K> lessThan;

    /**
     * Finds the node with the given key in the subtree of node curr,
     * or if there is none, the parent of where such a node would be.
     * @param key
     * @param curr The current node.
     * @param parent The parent of the current node.
     */
    Node<K> find(K key, Node<K> curr, Node<K> parent) {
```

**Solution:**

```
    Node<K> find(K key, Node<K> curr, Node<K> parent) {
        if (curr == null)   // 2 points
            return parent;
        else if (lessThan.test(key, curr.data)) // 3 points
            return find(key, curr.left, curr);
        else if (lessThan.test(curr.data, key)) // 3 points
            return find(key, curr.right, curr);
        else  // 2 points
            return curr;
    }
```

5. [6 points] Recall the definition of asympototically tight bound.

$$f \in \Theta(g) \text{ means } \exists k c_1 c_2, \forall n \geq k, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$$

Give an asymptotically tight bound for the following code.

```
sum = 0;
for (int k = 1; k <= n; k = k * 2) {
  for (int j = 1; j <= n; j = j * 2) {
    ++sum;
  }
}
```

**Solution:** The outer loop is $\log n$ iterations (2 points) and the inner loop is $\log n$ iterations (2 points), and the body of the inner loop is $\Theta(1)$, so the tight bound is $\Theta(\log n \log n)$ (2 points).

6. [8 points] Recall that $f \in O(g)$ if and only if $\exists kc.\forall n \geq k.f(n) \leq cg(n)$. Using this fact, prove that for any functions $f$ and $g$, if $f \in O(g)$ then $f + g \in O(g)$.

**Solution:** Because $f \in O(g)$, there exists $k_1$ and $c_1$ such that for any $n \geq k_1$,

$$f(n) \leq c_1 g(n). \qquad \text{(2 points)}$$

Choose $k = k_1$ (2 points). We need to show that for any $n \geq k_1$, exists $c$ such that

$$f(n) + g(n) \leq cg(n)$$

So we have

$$f(n) + g(n) \leq c_1 g(n) + g(n) \leq (c_1 + 1)g(n)$$

Choose $c = c_1 + 1$ (2 points).

So $f(n) + g(n) \leq cg(n)$ and therefore $f + g \in O(g)$ (2 points).

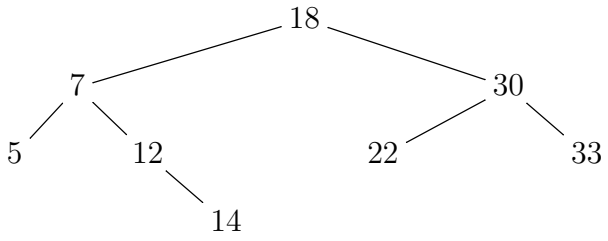
7. [10 points] The following `insert_sorted` method inserts the given `data` into a sorted linked list (sorted smallest-to-largest integer), producing a new sorted link list and leaving the original linked list unchanged. Fill in the blanks to complete `insert_sorted`.

```
class ListNode {
    int data;
    ListNode next;
    public ListNode(int val, ListNode next) {
        this.data = val; this.next = next;
    }
    public ListNode push(int val) {
      return new ListNode(val, this);
    }
    public ___(a)___ insert_sorted(int data) {
        if (data < this.data) {
            return this.push(___(b)___);
        } else if (this.next == null) {
            ListNode n = new ListNode(data, null);
            return n.push(___(c)___);
        } else {
            ListNode rest = ___(d)___;
            return rest.push(___(e)___);
        }
    }
}
```

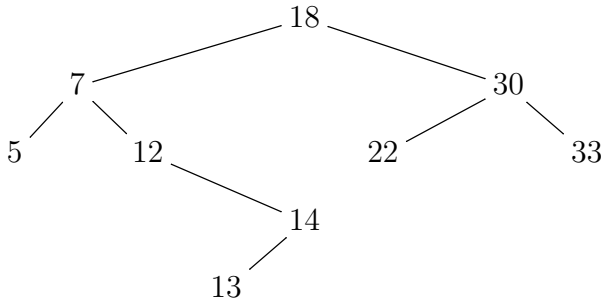
**Solution:** (2 points each)

```
a) ListNode
b) data
c) this.data
d) this.next.insert_sorted(data)
e) this.data
```

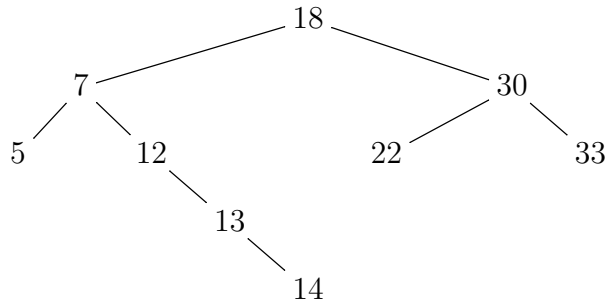
8. [8 points] Given the following AVL binary search tree, insert key 13, maintaining the binary search tree and AVL properties by using left and/or right rotations. Explain which nodes violate the AVL property and explain each change that you make to the tree. Draw the tree after each change.



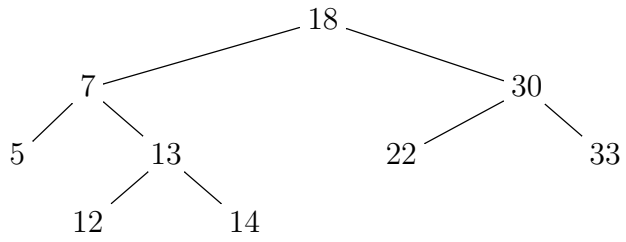

**Solution:** Insert 13 to the left of 14. (2 points)




Node 12 is not AVL. (2 points) Rotate right on 14. (2 points)




Rotate left on 12. (2 points)



9. [10 points] The `fast_exp` function implements exponentiation, so `fast_exp`$(x, n) = x^n$ for $n \geq 0$. Is the following formula a loop invariant for the `while` loop?

$$v \cdot y^m = x^n$$

Explain in detail why or why not, analyzing the body of the loop, statement by statement.

```
static int fast_exp(int x, int n) {
    if (n == 0) {
        return 1;
    }
    int v = 1, y = x, m = n;
    while (m != 1) {
        if (m % 2 == 0) {
            y = y * y;
            m = m / 2;
        } else {
            v = y * v;
            y = y * y;
            m = (m - 1) / 2;
        }
    }
    return y * v;
}
```

**Solution:** Yes, it is a loop invariant for the following reasons. (2 points)

1. The loop invariant is true before the start of the loop: $v \cdot y^m = x^n$ because $v = 1, y = x$, and $m = n$. (2 points)
2. For a hypothetical iteration of the loop, the loop invariant is true at the end of the loop body, assuming only that the loop invariant was true at the beginning of the loop body.

   Suppose $m$ is even, so the first branch is taken. We have $y^m = (y \cdot y)^{m/2}$, so the loop invariant $v \cdot y^m = x^n$ is equivalent to $v \cdot (y \cdot y)^{m/2} = x^n$. After the assignment `y = y * y`, we have $v \cdot y^{m/2} = x^n$. Then after the assignment `m = m / 2`, we have $v \cdot y^m = x^n$, which is the loop invariant. (2 points)

   Next we suppose $m$ is odd, so the second branch is taken. Note that $y^m = y \cdot y^{m-1}$. So the loop invariant is equivalent to $v \cdot y \cdot y^{m-1} = x^n$. After the assignment `v = y * v` we have $v \cdot y^{m-1} = x^n$. Observe that $m - 1$ is even, so $y^{m-1} = (y \cdot y)^{(m-1)/2}$. So we have $v \cdot (y \cdot y)^{(m-1)/2} = x^n$. After the assignment `y = y * y` we have $v \cdot y^{(m-1)/2} = x^n$. After the assignment `m = (m - 1) / 2` we have $v \cdot y^m = x^n$, which is the loop invariant. (2 points)
3. The loop invariant combined with the loop condition being false implies the correctness criteria for the `fast_exp` function. From the loop invariant $v \cdot y^m = x^n$ and $m = 1$ we have $v \cdot y = x^n$. Therefore `fast_exp`$(x, n) = x^n$. (2 points)

10. 8 points Give the big-O time complexity for each of the following methods of the `ArrayList` class.

    1. `E get(int index)`
    2. `boolean contains(Object o)`
    3. `boolean add(E e)`
    4. `E remove(int index)`

**Solution:** (2 points each)

1. $O(1)$
2. $O(n)$
3. $O(1)$
4. $O(n)$

11. 8 points Give the big-O time complexity for each of the following methods of the `LinkedList` class.

    1. `E get(int index)`
    2. `boolean contains(Object o)`
    3. `boolean add(E e)`
    4. `E remove()`

**Solution:** (2 points each)

1. $O(n)$
2. $O(n)$
3. $O(1)$
4. $O(1)$