CSE 373, Winter 2011
Midterm Key

1. Big-Oh (18 Points)

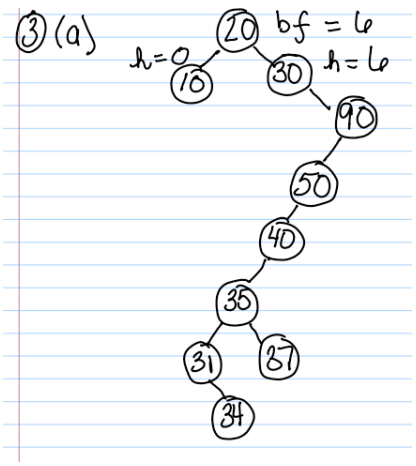
```
SUM (i = 1 to n + 3) { SUM (j = 1 to n^2) { 2 } - 1 }
SUM (i = 1 to n + 3) { 2 SUM (j = 1 to n^2) { 1 }  } - SUM (i = 1 to n + 3)  { 1 }
SUM (i = 1 to n + 3) { 2n^2 } - SUM (i = 1 to n + 3)  { 1 }
2n^2 (n + 3) - (n + 3)
2n^3 + 6n^2 - n - 3

O(n^3)
```

2. Sorting (12 Points)

Part	Conditions	Answer
a	array size 50000, random order	quick sort
b	array size 1000000, descending order, no extra memory may be allocated	heap sort
c	array size 350000, ascending order	insertion sort

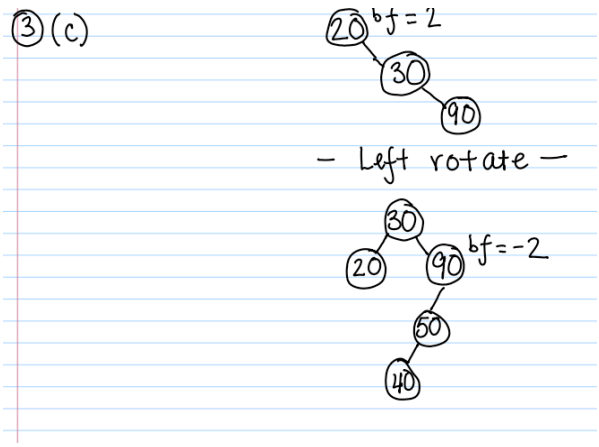
3. Trees (25 Points)

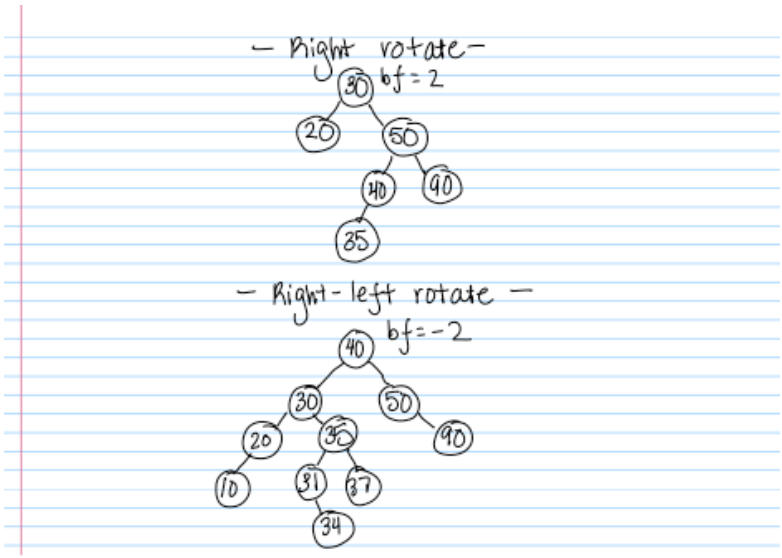
a.

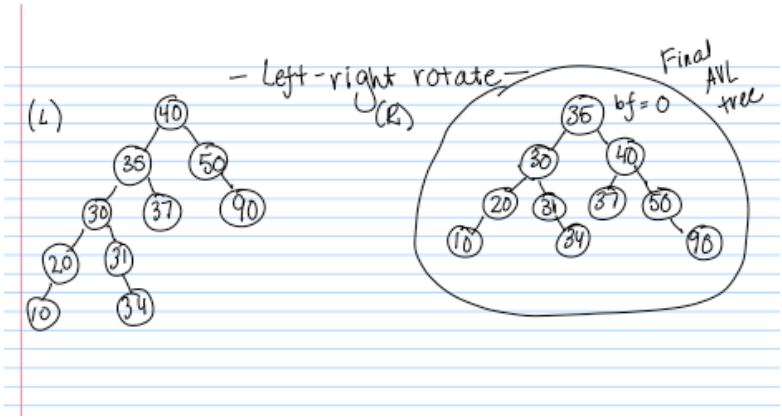


b. 6

c.



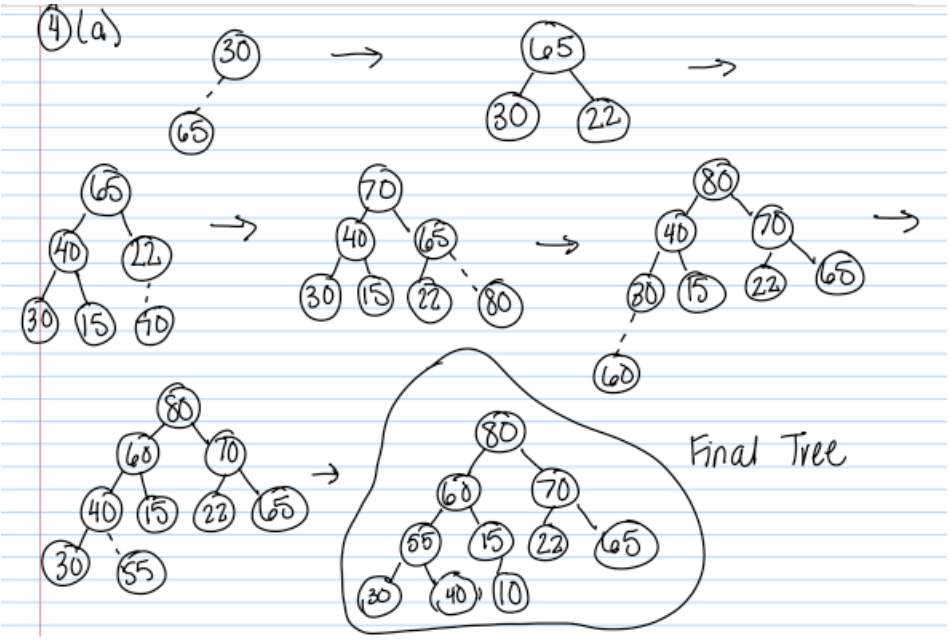

– Right rotate –
30
bf = 2
20
50
40
90
85
– Right - left rotate –
40
bf = -2
30
50
20
35
90
10
31
37
34


– Left - right rotate –
(L)
(R)
40
35
50
30
37
90
20
31
10
34
Final AVL tree
36
bf = 0
30
40
20
31
37
50
10
34
90

d. 0

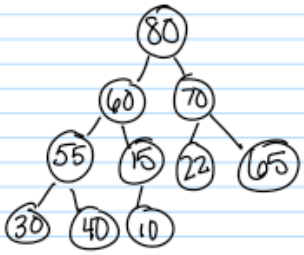
4. **Heaps** (20 Points)

a.

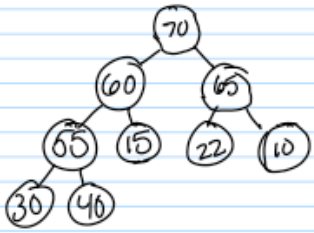


b.

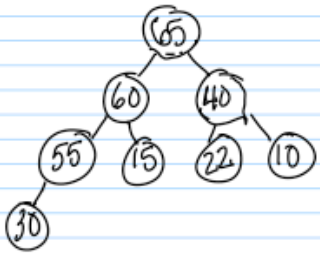
④(b)



remove #1



remove #2



5. Priority Queue Implementation

Part A: Implementation

Three solutions are shown:

```
public int remove()
{
    if (queue.isEmpty()) {
        throw new IllegalStateException();
    }

    // find min
    int min = queue.dequeue();
    queue.enqueue(min);

    for (int i = 1; i < queue.size(); i++) {
        int next = queue.dequeue();
        if (next < min) {
            min = next;
        }
        queue.enqueue(next);
     }

    // remove min
    int originalSize = queue.size();
    for (int i = 0; i < originalSize; i++) {
        int next = queue.dequeue();
        if (next != min) {
            queue.enqueue(next);
        }
    }
    return min;
}
```

```
 public int remove() {
    if (queue.isEmpty())
        throw new IllegalStateException();
    }

    int min = queue.dequeue();
    int size = queue.size();

    // examine each element
    // if element is less than min enqueue current min and reset
    // otherwise enqueue element
    for (int i = 0; i < size; i++) {
        int temp = queue.dequeue();
        if (temp < min) {
            queue.enqueue(min);
            min = temp;
        } else {
            queue.enqueue(temp);
        }
    }
    return min;
}
```

```
public int remove() {
    if (queue.isEmpty()) {
        throw new IllegalStateException();
    }

    int min = queue.dequeue();
    queue.enqueue(min);

    // find min
    for (int i = 0; i < queue.size() - 1; i++) {
        int next = queue.dequeue();
        if (next < min) {
            min = next;
        }
        queue.enqueue(next);
     }

    for (int i = 0; i < queue.size(); i++) {
        int next = queue.dequeue();
        if (min == next) {
            return min;
        }
        queue.enqueue(next);
    }
    return min;
}
```

Part B: Analysis

O(N)