Review Session Practice Midterm Solutions:

1.
mystery(7) = 7
mystery(825) = 258
mystery(38947) = 47893
mystery(612305) = 0523610
mystery(-12345678) = -785634120

2.
```
   public static int digitMatch(int a, int b) {
      if (a < 0 || b < 0) {
         throw new IllegalArgumentException();
      }

      int numMatch = 0;
      if (a % 10 == b % 10) {
         numMatch = 1;
      }

      if (a >= 10 && b >= 10) {
         numMatch = numMatch + digitMatch(a / 10, b / 10);
      }

      return numMatch;
   }
```

3.
```
   public static Map<String, Integer> wordCounts(List<String> l) {
      Map<String, Integer> wordsToCount = new TreeMap<>();

      for (String word : l) {
         if (!wordsToCount.containsKey(word)) {
            wordsToCount.put(word, 1);
         } else {
            int oldCount = wordsToCount.get(word);
            wordsToCount.put(word, oldCount + 1);
         }
      }

      return wordsToCount;
   }
```

4.

before	after	code

p->[1]->[2]
q->[3]
	p->[2]
	q->[3]->[1]
		q.next = p;
		p = p.next;
		q.next.next = null;

p->[1]
q->[2]->[3]
	p->[3]
	q->[1]->[2]
		p.next = q;
		q = p;
		p = p.next.next;
		q.next.next = null;

p->[1]->[2]
q->[3]->[4]
	p->[3]->[2]->[1]
	q->[4]
		p.next.next = p;
		ListNode temp = q.next;
		q.next = p.next;
		p.next = null;
		p = q;
		q = temp;

p->[1]->[2]
q->[3]->[4]->[5]
	p->[1]->[3]->[5]
	q->[2]->[4]
		p.next.next = q.next;
		q.next = q.next.next;
		ListNode temp = q;
		q = p.next;
		q.next.next = null;
		p.next = temp;

5.

```
public void removeFront(int n) {
   if (n < 0 || n > this.size) {
      throw new IllegalArgumentException();
   }

   for (int i = 0; i < size - n; i++) {
      this.elementData[i] = this.elementData[i + n];
   }

   this.size = this.size - n;
}
```

6.

```
public void rearrangeDuplicates(Queue<Integer> q) {
   if (q.size() > 0) {
      Stack<Integer> s = new Stack<>();
      int size = q.size();
      int v = q.remove();
      q.add(v);
      for (int i = 1; i < size; i++) {
         int newValue = q.remove();
         if (newValue == v) {
            s.push(newValue);
         } else {
            v = newValue;
            q.add(v);
         }
      }

      int stackSize = s.size();
      while (!s.isEmpty()) {
         q.add(s.pop());
      }

      for (int i = 0; i < size - stackSize; i++) {
         q.add(q.remove());
      }

      for (int i = 0; i < stackSize; i++) {
         s.push(q.remove());
      }

      while (!s.isEmpty()) {
         q.add(s.pop());
      }
   }
}
```