Solution to CSE143 Section #3 Problems

1. One possible solution appears below.

```
public String acronymFor(List<String> words) {
    String acronym = "";
    for (String next : words) {
        acronym += next.charAt(0);
    }
    return acronym.toUpperCase();
}
```

2. Two possible solutions appear below.

```
public void switchPairs(List<String> list) {
    for (int i = 0; i < list.size() - 1; i += 2) {
        String first = list.remove(i);
        list.add(i + 1, first);
    }
}
```

```
public void switchPairs(List<String> list) {
    int i = 0;
    while (i < list.size() - 1) {
        String first = list.get(i);
        list.set(i, list.get(i + 1));
        list.set(i + 1, first);
        i += 2;
    }
}
```

3. One possible solution appears below.

```
public void stutter(List<Integer> list) {
    for (int i = 0; i < list.size(); i += 2) {
        list.add(i, list.get(i));
    }
}
```

4. Two possible solutions appear below.

```
public void reverse3(List<Integer> list) {
    for (int i = 0; i < list.size() - 2; i += 3) {
        int n1 = list.get(i);
        int n3 = list.get(i + 2);
        list.set(i, n3);
        list.set(i + 2, n1);
    }
}
```

```
    public void reverse3(List<Integer> list) {
        for (int i = 0; i < list.size() - 2; i += 3) {
            list.add(i, list.remove(i + 2));
            list.add(i + 2, list.remove(i + 1));
        }
    }
```

5. One possible solution appears below.

```
    public boolean hasOdd(Set<Integer> set) {
        for (int value : set) {
            if (value % 2 != 0) {
                return true;
            }
        }
        return false;
    }
```

6. One possible solution appears below.

```
    public Set<Integer> removeEvens(Set<Integer> s) {
        Set<Integer> result = new TreeSet<>();
        Iterator<Integer> i = s.iterator();
        while (i.hasNext()) {
            int n = i.next();
            if (n % 2 == 0) {
                result.add(n);
                i.remove();
            }
        }
        return result;
    }
```

7. One possible solution appears below.

```
    public boolean containsAll(Set<Integer> s1, Set<Integer> s2) {
        Iterator<Integer> i = s2.iterator();
        while (i.hasNext()) {
            if (!s1.contains(i.next())) {
                return false;
            }
        }
      return true;
    }
```

8. One possible solution appears below.

```
    public boolean equals(Set<Integer> s1, Set<Integer> s2) {
        if (s1.size() != s2.size()) {
```

```
            return false;
        }
        for (int n : s1) {
            if (!s2.contains(n)) {
                return false;
            }
        }
        return true;
    }
```

9. One possible solution appears below.

```
    public void retainAll(Set<Integer> s1, Set<Integer> s2) {
        Iterator<Integer> i = s1.iterator();
        while (i.hasNext()) {
            if (!s2.contains(i.next())) {
                i.remove();
            }
        }
    }
```