CSE 373 Practice Midterm Exam #2
ANSWER KEY

1. Big-Oh Analysis

a) O(N^4)
b) O($(\log N)^2$)
c) O($N \log N$)
d) O($N \log N$)
e) O(N^2)
f) O(N)

2. Java / Guava Collection Programming

```
public static boolean friends(Multiap<String, String> map, List<String> names) {
    for (int i = 0; i < names.size(); i++) {
        String n1 = names.get(i);
        for (int j = i + 1; j < names.size(); i++) {
            String n1 = names.get(j);
            if (!map.get(n1).contains(n2) ||
                !map.get(n2).contains(n1)) {
                return false;
            }
        }
    }
    return true;
}
```

3. Java Class Programming for Collections

```
public class Person {
    private String name;
    private String gender;
    private Person fiancee;
    private Queue<String> preferences;

    ...

    public boolean equals(Object o) {
        if (o instanceof Person) {
            Person other = (Person) o;
            return name.equals(other.name) && gender.equals(other.gender)
                && fiancee == other.fiancee
                && preferences.equals(other.preferences);
        } else {
            return false;
        }
    }

    public int hashCode() {
        return  13 * name.hashCode() +
                37 * gender.hashCode() +
               117 * Boolean.valueOf(fiancee != null).hashCode() +
               313 * preferences.hashCode();
    }
}


public class PersonComparator implements Comparator<Person> {
    public int compare(Person p1, Person p2) {
        if (!p1.gender.equals(p2.gender)) {
            return -p1.gender.compareTo(p2.gender);
        } else {
            return p1.name.compareTo(p2.name);
        }
    }
}
```

4. Hashing

```
   +---+
0  | / |
   +---+
1  | / |--> 31=17
   +---+
2  | / |--> 72=5 --> 2=3
   +---+
3  | / |
   +---+
4  | / |
   +---+
5  | / |
   +---+
6  | / |
   +---+
7  | / |
   +---+
8  | / |
   +---+
9  | / |
   +---+

size        =  3
capacity    = 10
load factor =  0.3
```

5. Heaps

a) after all adds, final min-heap tree:

```
              10
           /      \
      15              12
    /    \          /    \
  43      17      40      13
 /  \    /  \
95  47  63  82
```

array:

```
 0   1   2   3   4   5   6   7   8   9  10  11  12  13
[/, 10, 15, 12, 43, 17, 40, 13, 95, 47, 63, 82,  /, ...]
```

b) after 2 removes, final min-heap tree:

```
              13
           /      \
      15              40
    /    \          /    \
  43      17      63      82
 /  \
95  47
```

array:

```
 0   1   2   3   4   5   6   7   8   9  10  11
[/, 13, 15, 40, 43, 17, 63, 82, 95, 47,  /, ...]
```

6. AVL Trees

a) after all adds, AVL tree:

```
                    42
                /        \
           30                48
         /    \            /    \
      20      40         45      65
     /       /          /  \    /  \
   10       35         43  47  51  73
```

b) after all removes, AVL tree:

```
             48
          /      \
       20           65
      /  \         /  \
    10    35     51    73
```

7. Heap Priority Queue Implementation

```
public void reverse() {
    int i1 = front;
    int i2 = (front + size - 1) % elements.length;
    for (int i = 0; i < size / 2; i++) {
        int temp = elements[i1];
        elements[i1] = elements[i2];
        elements[i2] = temp;
        i1 = (i1 + 1) % elements.length;
        i2 = (i2 - 1 + elements.length) % elements.length;
    }
}
```