Partial solution

Question 6. [10 points] For each of the following code fragments (a)–(d), state a big-O upper bound on the running time, with the problem size n being the value of the variable `n`. Briefly explain each bound.

(a)

```
int sum = 0;
for (int j = 0; j < n; j++) {        — n times
    for (int i = 0; i < n; i++) {    — n times
        sum++;  — O(1)
    }
}
```

$n \cdot n \cdot O(1)$ is $O(n)$

(b)

```
int sum = 0;
for (int j = 0; j < n; j++) {        — n times
    int max = j;
    if (max > 100) { max = 100; }
    for (int i = 0; i < max; i++) {  — at most 100 times
        sum++;  O(1)
    }
}
```

$n \cdot 100 \cdot O(1)$ is $O(n)$

(c)

```
int sum = 0;
for (int j = 0; j < n*n; j++) {      — n^2 times
    for (int i = 0; i < j; i++) {    0, 1, 2, 3, ..., n^2-2, n^2-1 times
        sum++;
    }
}
```

$$\sum_{i=0}^{n^2-1} i = \overbrace{0 + \underbrace{1 + 2 + \dots + (n^2-2)}_{n^2-1} + (n^2-1)}^{n^2-1}$$

pairs sum to (n^2-1)

$$= \frac{n^2}{2} \cdot (n^2-1) = \frac{n^4}{2} - \frac{n^2}{2}$$

which is $O(n^4)$

↑ # of pairs

(d)

```
int sum = 0;
for (int j = 0; j < n; j++) {         ← n times
    for (int i = 0; i < n; i = i*2) { ← at most log_2 n times
        sum++;
    }
}
```

$O(n \cdot \log n)$

Question 7. [10 points] Complete the following method. It should remove all elements of the given `List` that compare as greater than the `val` parameter according to the comparator object given as the `comp` parameter. Note that the method *should* modify `list` (by removing elements).

```
public static<E>
void removeAllGreaterThan(List<E> list, E val, Comparator<E> comp) {

    Iterator<E> i = list.iterator();

    while (i.hasNext()) {
        E elt = i.next();
        if (comp.compare(elt, val) > 0) {
            i.remove();
        }
    }

}
```