CS 201, Fall 2016 — Dec 2nd — Exam 3 Name: Solution

Question 1. [6 points] State a big-O upper bound on the worst-case running time of the following method, where the problem size N is the number of elements in the `ArrayList` passed as the parameter. Briefly justify your bound.

```
public static<E extends Comparable<E>>
Collection<E> getSortedCollection(ArrayList<E> list) {
    TreeSet<E> treeSet = new TreeSet<E>();
    for (E elt : list) {          — N times
        treeSet.add(elt);         O(log N)
    }
    return treeSet;
}
```

O(N log N) because there are N elements in the ArrayList and the cost of adding an element to a TreeSet is O(log N)

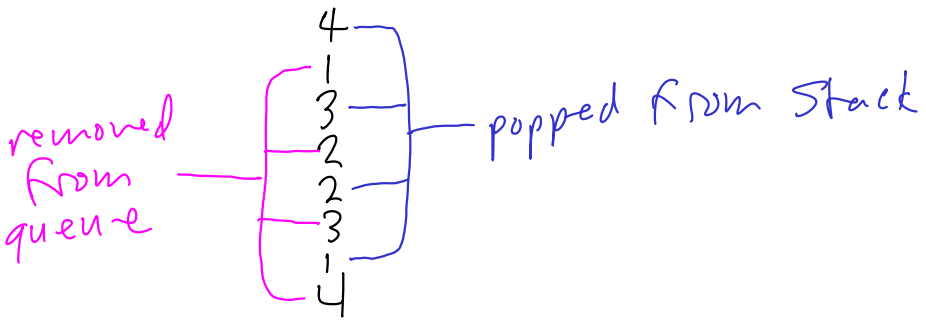
Question 2. [6 points] State a big-O upper bound on the worst-case running time of the following method, where the problem size N is the number of elements in the `LinkedList` passed as a parameter. Briefly justify your bound.

```
public static void removeEven(LinkedList<Integer> list) {
    Iterator<Integer> i = list.iterator();
    while (i.hasNext()) {         — N times
    Integer val = i.next();       O(1)
        if (val % 2 == 0) {
            i.remove();           O(1)
        }
    }
}
```

O(N) because there are N elements in the LinkedList, calling next() on the iterator is O(1) and calling remove() on the iterator is also O(1)

Question 3. [6 points] What output is printed by the following code?

```
Queue<Integer> q = new LinkedList<Integer>();
Stack<Integer> s = new Stack<Integer>();

for (int i = 1; i <= 4; i++) {
    q.add(i);
    s.push(i);
}
for (int i = 1; i <= 4; i++) {
    System.out.println(s.pop());
    System.out.println(q.remove());
}
```



Question 4. [6 points] Consider the following recursive method, which is intended to compute the sum of the integers from 1 to `n`:

```
public static int sumInts(int n) {
    return sumInts(n-1) + n;
}
```

(a) Briefly explain the bug in this method.

There is no base case, so the recursion will not terminate.

(b) What exception will be thrown by the method when it is called?

StackOverflowError

(c) Briefly describe how to fix the bug.

Check the base case on entry to the method:

```
if (n == 1) {
    return 1;
}
```

Question 5. [8 points] Consider the following recursive method:

```
public static int pigSeq(int sum) { // assume sum is non-negative
    if (sum == 0) { return 1; }
    if (sum == 1) { return 0; }
    int numSeq = 0;
    for (int i = 2; i <= 6; i++) {
        if (sum - i >= 0) { numSeq += pigSeq(sum - i); }
    }
    return numSeq;
}
```

Write a memoized version of this function called `pigSeqMemo`. It will be called by the following function:

```
public static int pigSeq(int sum) {
    return pigSeqMemo(sum, new int[sum + 1]);
}
```

Hints:

- The original version of `pigSeq` always returns a non-zero value for any value of `sum` 2 or greater
- The array passed to `pigSeqMemo` can be indexed by any integer 0..`sum`, inclusive
- Make sure that a recursive call to `pigSeqMemo` is only made once per sub-problem

```
public static int pigSeqMemo(int sum, int[] memo) {
    if (sum == 0) { return 1; }
    if (sum == 1) { return 0; }
    int ans = memo[sum];
    if (ans == 0) {
        int numSeq = 0;
        for (int i = 2; i <= 6; i++) {
            if (sum - i >= 0) {
                numSeq += pigSeqMemo(sum-i, memo);
            }
        }
        ans = numSeq;
        memo[sum] = ans;
    }
    return ans;
}
```

Question 6. [6 points] Briefly explain under what circumstances adding additional threads will likely not speed up (and could even slow down) the performance of an algorithm that can be divided into N independent tasks.

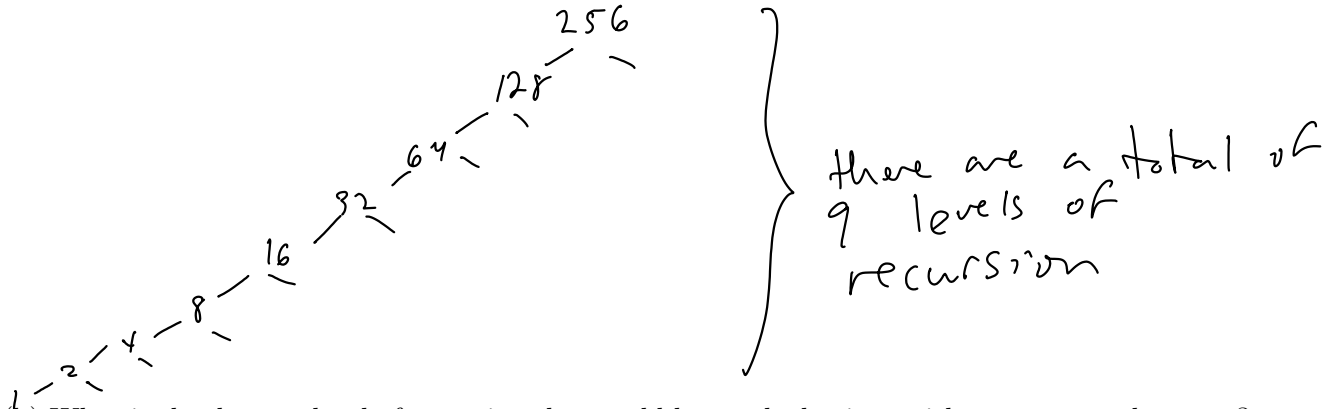
- Using more threads than cores will tend not to result in faster execution
- However, if it is difficult to allocate equal amounts of work to each task, then a larger number of small tasks may yield better overall performance

Question 7. [6 points] Complete the following code to update the inventory count for a given make of car. Note: There are 3 separate lines of code to fill in.

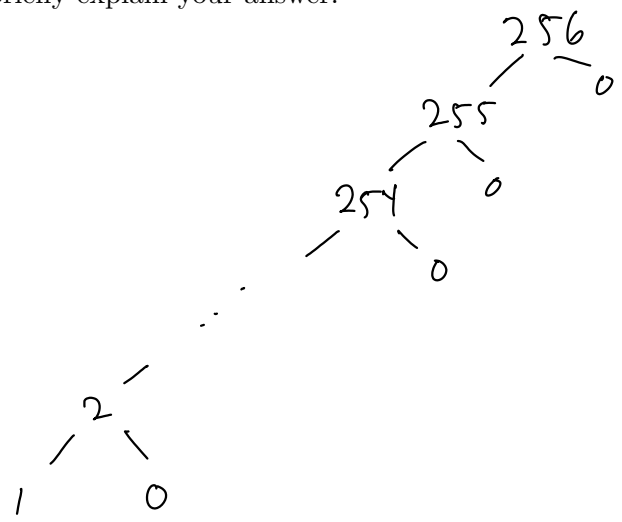
```
public Integer updateCarInventory(
    TreeMap<String, Integer> carInventory, String car, Integer count) {

    // if car is not in the inventory...
    if (!carInventory.containsKey(car)) {
        // ...add it to the map (add CODE here)
        carInventory.put(car, count);
    }
    else {
        // otherwise, update the car's count in the map (add CODE here)
        int prev = carInventory.get(car);
        carInventory.put(car, prev + count);
    }
    // return the updated count for the car (add CODE here)
    return carInventory.get(car);
}
```

Question 8. [6 points] Consider an array of 256 (2^8) integers:

(a) What is the deepest level of recursion that could be reached using merge sort to sort the array? (Hint: recall that merge sort works by dividing ranges in half and recursively sorting them.) In order to receive any partial credit, briefly explain your answer.



(b) What is the deepest level of recursion that could be reached using quick sort to sort the array? (Hint: recall that there is an unlikely $O(N^2)$ special case.) In order to receive any partial credit, briefly explain your answer.



If every partition creates one completely empty partition, then only one element is eliminated by each recursive call, resulting in 257 levels of recursion. (This is extremely unlikely if a reasonable pivot selection strategy is used.)

Programming Questions

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

Important: You may use the following resources:

- The textbook
- The lecture notes posted on the course web page
- Your previous labs and assignments

Do not open any other files, web pages, etc.

Question 9. [25 points] A "tall" character is one of the lower-case letters b, d, f, h, k, l, or t, or an upper-case letter. Implement the `countTall` method so that it returns the number of "tall" characters in the string passed as the parameter. **Important**: you *must* use recursion. Do not use a loop.

Hints:

- Think about an appropriate base case
- Think about how to find a subproblem
- Think about how to take the recursive solution to the subproblem and extend it to be a solution to the overall problem
- `s.charAt(i)` returns the character at index `i` in the string `s`
- `s.substring(start, end)` returns the substring of `s` from index `start` (inclusive) to index `end` (exclusive)
- `s.substring(start)` returns the substring of `s` from index `start` (inclusive) to the end of the string
- `Character.isUpperCase(c)` returns true if a character `c` is an upper case letter, false otherwise

Unit tests are provided in `Q9Test`. Make sure the unit tests pass.

Question 10. [25 points] Complete the implementation of the `Stats` class. The idea is that each time the `updateStats` method is called, information is being provided about the performance of one player in one particular softball game, specifically the player's name (`player`), the game number (`game`), and the number of runs scored by that player (`numRuns`). The `updateStats` method should keep track of the total number of runs by each player (over all games), and the total number of runs in each game (over all players).

Once a `Stats` object has been populated with data, the `getRunsForPlayer` and `getRunsForGame` methods can be used to get the total number of runs for a particular player or game, respectively.

The `stats.csv` file contains the raw data. The `Q10.readStats` method creates a new `Stats` object, reads the data from `stats.csv`, and calls `updateStats` for each data record in the file. You should not modify either `stats.csv` or `Q10`, but you may find it helpful to refer to them. (To open `stats.csv`, right click and choose Open with → Text editor.)

The `Q10Test` class has unit tests. Make sure all of the unit tests pass.

Hints:

- Use a map of `String` (player) to `Integer` (runs) for keeping track of the number of runs for each player
- Use a map of `Integer` (game) to `Integer` (runs) for keeping track of the number of runs for each game
- `updateStats` should create or update one entry in each map
- The `getRunsForPlayer` and `getRunsForGame` methods should look up the desired number of runs in the appropriate map