CSE 332: Data Abstractions — Autumn 2015

Final Exam

Name:			
ID #:			
TA:		**Section:**	

INSTRUCTIONS:

- You have **110 minutes** to complete the exam.
- The exam is closed book. You may not use cell phones or calculators.
- All answers you want graded should be written on the exam paper.
- If you need extra space, use the back of a page.
- The problems are of varying difficulty.
- If you get stuck on a problem, move on and come back to it later.
- It is to your advantage to read all the problems before beginning the exam.

Problem	Points	Score	Problem	Points	Score
1	13		5	10	
2	10		6	12	
3	10		7	20	
4	10		8	15	
			Σ	100	

One Liners.

This section has questions that require very short answers. To get full credit, you should answer in no more than one sentence per question.

1. ASN [13 points]
For each of the following, answer **ALWAYS**, **SOMETIMES**, or **NEVER**. Give a brief (one sentence) explanation of your answer for each.

(a) (2 points) In an implementation of the `UnionFind` ADT, `find` is $\mathcal{O}(1)$.

(b) (2 points) In an implementation of the `UnionFind` ADT, `union` is amortized $\mathcal{O}(1)$.

(c) (2 points) An algorithm that solves the SORT problem must have $\Omega(n \lg n)$ swaps.

(d) (2 points) `SORT` $\in$ P

(e) (2 points) `CIRCUITSAT` $\in$ NP

(f) (2 points) `BST-FIND` $\in$ NP (Given a BST T and a number n, is $n \in T$?)

(g) (1 point) Radix Sort can be used to sort a list of numerical data.

Basic Techniques.

This part will test your ability to apply techniques that have been explicitly identified in lecture and reinforced through sections and homeworks. Remember to show your work and justify your claims.

2. RCTOA NDECOIIN [10 points]

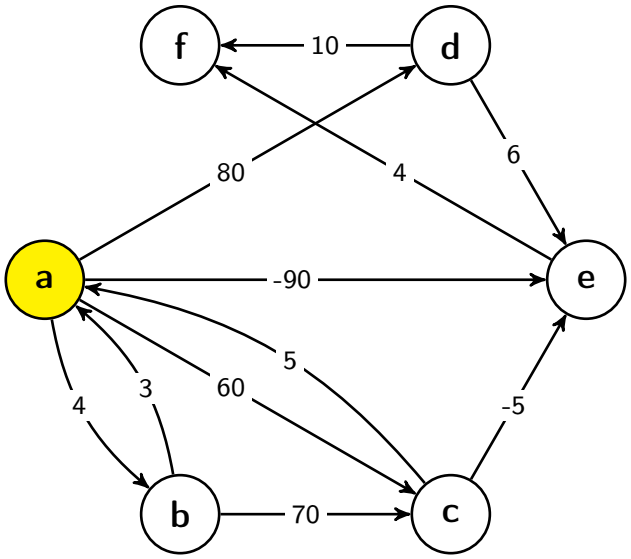
Imagine we run `main`. Is there a race condition? If there is one, explain why by showing a bad interleaving and explaining what the race is. If not, explain why not.

```
1  public static Lock lock = new ReentrantLock();
2  public static Stack<Integer> stack;

3  public static void main(String[] args) {
4     stack = <initialize stack with elements>;
5
6     Task t1 = <run task>;
7     Task t2 = <run task>;
8     t1.fork();
9     t2.fork();
10    int sum = t1.join() + t2.join();
11    System.out.println("sum = " + sum);
12 }
```

```
13 public int task() {
14    int count = 0;
15    while (!stack.isEmpty()) {
16       lock.lock();
17       count += stack.pop();
18       lock.unlock();
19    }
20    return count;
21 }
```

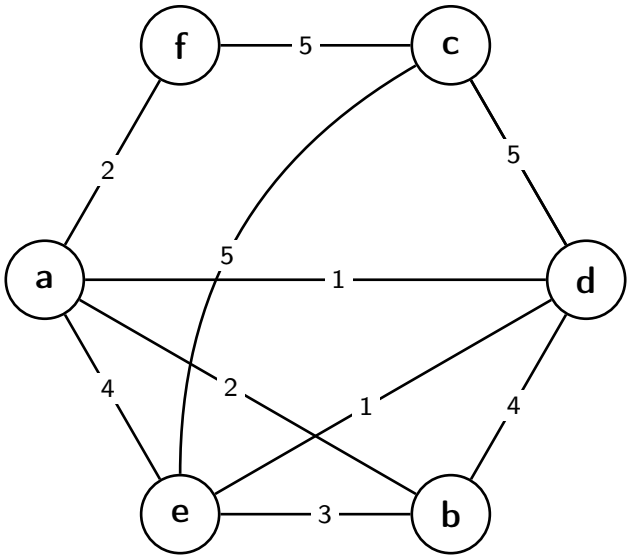

3. GOTO Considered Harmful; while loops considered okay [10 points]



(a) (6 points) Use Dijkstra's Algorithm to find the **lengths** of the shortest paths from **a** to each of the other vertices. For full credit, you must show the worklist at every step, but how you show it is up to you.

(b) (4 points) Are any of the lengths you computed using Dijkstra's Algorithm in part (a) incorrect? For each length that is incorrect, explain what the correct answer is and why the answer from part (a) was incorrect.

4. Spring Time! [10 points]



(a) (5 points) Use Kruskal's Algorithm to find *two* minimum spanning trees of the above graph.

(b) (5 points) Imagine that the above graph had some negative edges in it. Would Prim's Algorithm necessarily return a correct result? Explain your answer in 1-2 sentences.

5. Definitely A Graph! [10 points]

(a) (5 points) Suppose you are given a graph G. Explain how you would figure out if it has a cycle.

(b) (5 points) Suppose you are given a DAG G representing the work graph of a bunch of `ForkJoin` tasks. Explain how you would determine the longest dependency path of G. What does the longest dependency path in G represent with respect to the algorithm G?

6. X-Tra [12 points]
Consider the `expand` problem. `expand` takes in an `int[]` and outputs a new `int[]` where each element `a[x]` is copied `a[x]` times. For example,

```
expand([1, 2, 1, 4, 5, 3]) = [1, 2, 2, 1, 4, 4, 4, 4, 5, 5, 5, 5, 5, 3, 3, 3]
```

Give a high-level algorithm to solve the `expand` problem. DO NOT WRITE CODE! Your solution should have $\mathcal{O}(n)$ work and $\mathcal{O}(\lg n)$ span where n is the sum of the elements in the array. Be sure to explain why your solution meets this bound.

A Moment's Thought!

This section tests your ability to think a little bit more insightfully. The approaches necessary to solve these problems may not be immediately obvious. Remember to show your work and justify your claims.

7. Peek-a-boo [20 points]

(a) (15 points) Write a `ForkJoin` algorithm to solve the following problem:

Input(s): An int k, An array of ints with values between 0 and k **Output**: The largest missing number in the array, or -1 if none of them is missing

Your solution must have $\mathcal{O}(n)$ work and $\mathcal{O}(\lg n)$ span where n is the size of the input array. You may assume that k is much smaller than n. You may not use any global data structures or synchronization primitives (locks).

```
public class LargestMissingNumber {
   private static final ForkJoinPool POOL = new ForkJoinPool();

   public int largestMissingNumber(int k, int[] input) {




   }
```

```
}
```

(b) (5 points) Write recurrences for the work and the span for your solution in terms of n and k.

8. FootPen Needs Your Help! [15 points]

A new social networking company called FootPen has arrived on the start-up scene. Because UW CSE students are known to be the best in industry, FootPen has asked you to help them implement several features. For each feature request,

- explain how to represent the problem as a graph,
- give a (high-level) idea for an algorithm to solve the problem, and
- give a runtime analysis of your algorithm.

(a) (5 points) How can FootPen determine the number of people that have at least one friend in common with a particular user.

(b) (5 points) FootPen would like to add a "make a new friend" feature. To facilitate this feature, FootPen has an algorithm that generates an "familiarity score" for every pair of friends. The familiarity score is a real number between 0.0 and 1.0 which represents how well two users know each other. The closer the score is to zero, the more friendly the two users are.

The "make a new friend" feature is intended to help users find *a single user* who *they do not know* but are likely to get along with. We can estimate a familiarity score between two users who *are not friends* by summing the scores of the shortest weight chain of friends that joins the two non-friends.

FootPen has run experiments and found that an estimated familiarity score of between 0.5 and 1 is likely to indicate that two users do not already know each other but would make good friends.

How can FootPen write the "make a friend feature"?

(c) (5 points) Now that FootPen has become popular, the government has sent a request asking for the group of people who correspond with each other the most. FootPen has records the number of conversations each pair of users has in its database. How can FootPen identify the k users with the largest number of inter-communications?