CSE 332 Midterm Exam

Data Structures and Parallelism

University of Washington, Winter 2020

This exam has 5 questions worth a total of 40 points and is to be completed in 50 minutes. Two double-sided, 8.5-by-11" sheets of notes are permitted. Electronic devices are prohibited. This exam is preprocessed by a computer when grading, so please **write darkly and write your answers inside the designated spaces. Write the statement below in the given blank and sign. You may do this before the exam begins.**

> "I have neither given nor received any assistance in the taking of this exam."

Signature: _____

Question	Points
1	1
2	13
3	15
4	4
5	7
Total	40

Name	
Student ID	
Name of person to left	
Name of person to right	

- Take notes in any white space. **Only answers written in the designated spaces will be graded.**
- ◯ indicates that only one circle may be chosen. Fill-in the shape completely: ●.
- ☐ indicates that more than one box may be selected. Fill-in the shape completely: ■.
- Work through the problems you are comfortable with first.
- Not all information provided in a problem may be useful.
- Unless we specifically give you the option, the correct answer is not, 'does not compile.'

Designated Exam Relaxation Space

1. (1 pt) **So It Begins** Write the statement on the front page and sign. Write your name, ID, and the names of your neighbors. Write your name in the given blank in the corner of every other page. Enjoy a free point.

2. (13 pts) **Algorithm Analysis** Give the worst-case order of growth of the runtime in $\Theta(\cdot)$ notation as a function of N. Give a simple answer with no unnecessary leading constants or summations.

(a) Θ(_____)

```
static void fastMult(int N) {
    if (N <= 1) { return N * N; }
    for (int i = N; i > 0; i -= 1000) {
        if (i < 10000) { return N * N * N; }
    } }
```

(b) Θ(_____)

```
static void roll(int N) {
    if (N > 0) {
        for (int i = 0; i < N; i += 2) {
            System.out.println(i);
        }
        roll(N / 2);
    } }
```

(c) Θ(_____)

```
static void loopy(int N) {
    int Q = 0;
    for (int i = 1; i < N; i *= 2) { Q += 1; }
    for (int i = 1; i < Q; i += 1) {
        for (int j = 0; j < i; j += 1) {
            System.out.println(j);
        }
    } }
```

(d) Θ(_____)

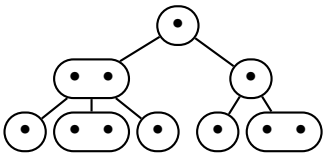
```
static void countdown(int N) {
    if (N > 0) {
        countdown(N / 2);
        countdown(N / 2);
        for (int i = 1; i <= N; i += 1) {
            for (int j = i; j > 0; j -= 1) {
                System.out.println(j);
            }
        }
    } }
```

(e) Suppose we know that the order of growth of a function is in $\Omega(\log N)$, $O(N)$. Select all true expressions.

☐ $O(1)$	☐ $\Theta(1)$	☐ $\Omega(1)$
☐ $O(\log N)$	☐ $\Theta(\log N)$	■ $\Omega(\log N)$
■ $O(N)$	☐ $\Theta(N)$	☐ $\Omega(N)$
☐ $O(N \log N)$	☐ $\Theta(N \log N)$	☐ $\Omega(N \log N)$
☐ $O(N^2)$	☐ $\Theta(N^2)$	☐ $\Omega(N^2)$

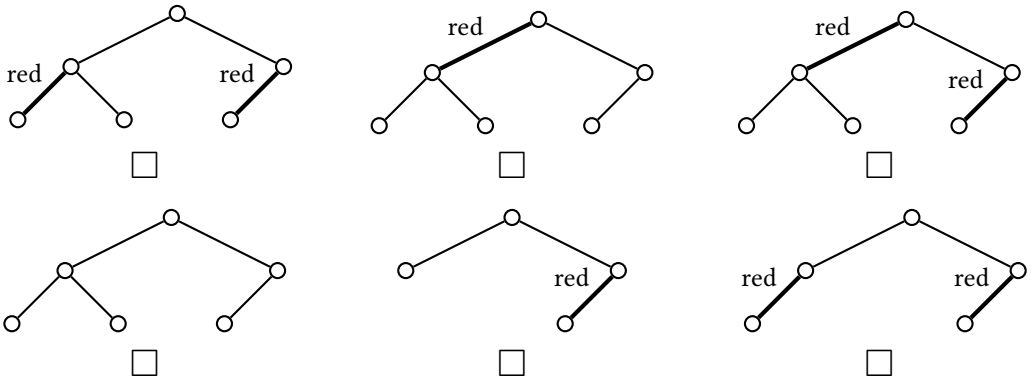
3. (15 pts) **Trees**

(a) What is the maximum number of keys that can be added to this 2-3 tree without increasing its height?

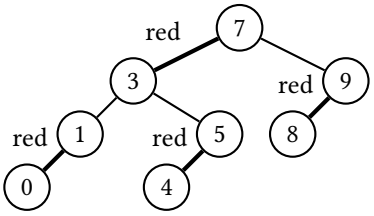


Additional keys: _____

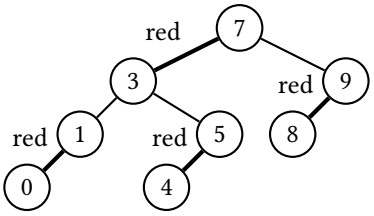
(b) Select all valid left-leaning red-black tree structures.



(c) Draw the corresponding 2-3 tree after inserting 2. Do not draw a left-leaning red-black tree.

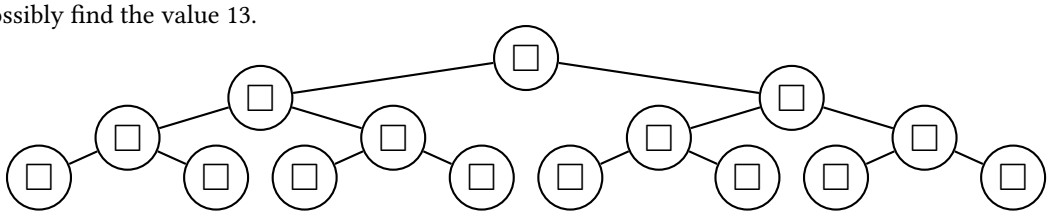


(d) What left-leaning red-black tree operations are executed when inserting 2? In the order of execution, choose one operation per row and fill in the node value. You may not need all rows.

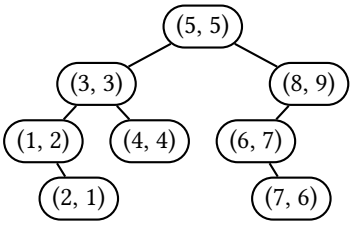


1. `rotateLeft(____)` `rotateRight(____)` `flip(____)`
2. `rotateLeft(____)` `rotateRight(____)` `flip(____)`
3. `rotateLeft(____)` `rotateRight(____)` `flip(____)`
4. `rotateLeft(____)` `rotateRight(____)` `flip(____)`
5. `rotateLeft(____)` `rotateRight(____)` `flip(____)`

(e) If the integers 1–15 are inserted into a max-heap in an unknown order, select all nodes where we could possibly find the value 13.



(f) The following is a k-d tree, but it is also a valid search tree according to other partitioning rules. Select all valid partitioning rules. Assume y-coordinate levels partition down to the left and up to the right.



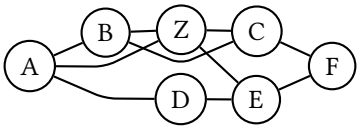
Level 0	x	y	x	x	y	y
Level 1	y	x	x	x	y	y
Level 2	x	y	x	y	y	x
Level 3	y	x	x	y	y	x
	■	☐	☐	☐	☐	☐

4. (4 pts) **Hashing** Say we implement `PointSet.nearest` with a hash table storing points by x-coordinate, a non-negative decimal value. The bucket index is given by rounding the x-coordinate to the nearest integer.

(a) Describe an algorithm for `nearest(Point target)` that checks as few hash table buckets as possible.

(b) Describe a collection of input points that results in worst-case runtime for `nearest` given any target.

5. (7 pts) **Graphs** Consider this undirected graph. When traversing the graph, visit neighbors alphabetically.



(a) Starting at A: the last vertex on the DFS path to Z is _____; the last vertex on the BFS path to F is _____.

(b) Given a connected undirected graph $G = (V, E)$, describe an algorithm to find a vertex whose removal does not disconnect the graph in $O(|V| + |E|)$ time. Briefly justify correctness using graph terminology.