SER 222 **Practice** Exam 3

Updated 2/19/2019

Last Name: ______________________
First Name: ______________________
Last 4 digits of ASU ID: ______________

Exam Instructions

The exam is open textbook (Algorithms 4e by Sedgewick and Wayne), as well as open note. No electronic items are allowed. Write legibly. Please use a pen (instead of a pencil) if you have one. There are 90 points available and the exam must be completed in 37.5 minutes. This exam has two types of questions:

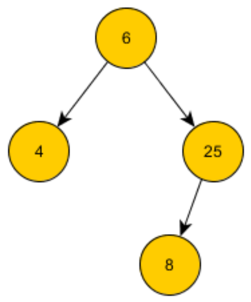
Short answer questions: There are 90 points of short answer questions. A typical answer is one or two sentences. Each short answer question is worth 5 or 10 points.

Write-in questions: The programming questions are given near the end of the paper. They must be answered on the question paper. There are 20 points of longer write-in questions.

Topic	**Earned**	**Possible**
SA: Binary Search Trees		20
SA: Hashtables		30
SA: Undirected Graphs		20
SA: Directed Graphs & SP		20
Scen: Hashtables		0
Scen: Undirected Graphs		20
Scen: Directed Graphs & SP		0
Total:		100

Short Answer: Binary Search Trees

1. Trace a (non-balancing) BST through the following operations:



```
//assume that the "tree" variable below has already been construct
//to be the tree shown above.
tree.add(new Integer(3));
tree.add(new Integer(7));
tree.add(new Integer(2));
tree.remove(new Integer(6));
```

Draw the structure of the BST after each call. Images must be cumulative. [10 points]

2. When adding and removing nodes from a heap, the binary structure will always remain balanced. Why is it the case that when these operations are performed over a BST, there is a a chance that the tree will become unbalanced? (Hint: array versus list representation doesn't matter.) [10 points]

Short Answer: Hashtables

3. Is the hashCode() implementation in the following Point2D class valid? That is, would a hashtable be able to properly store keys with this hashCode? Explain. [10 points]

```
public class Point2D {
    public int x, y;

    public Point2D(int x, int y) { this.x = x; this.y = y;}

    public int hashCode() {
        return x % Integer.MAX_VALUE;
    }
}
```

4. Imagine that you have a linear probe hashtable, with M=11. Draw the final hashtable after adding these keys: 34, 13, 10, 3, 0, 15, 14. Use the hash function $hash(k, i) = (k \ mod\ 11 + i)\ mod\ 11$, where i is number of times the algorithm has tried to insert the key. Your drawing should include the main size M array. [20 points]

Short Answer: Undirected Graphs

5. Would the DFS algorithm run faster if it was implemented with an adjacency matrix or an adjacency list? [10 points]

6. If you are asked to run a DFS or BFS by hand, and given both a picture of the graph and the code for the algorithm, why is there a chance you and a classmate who correctly execute the algorithm will produce different paths? [10 points]

Short Answer: Directed Graphs

7. When using DFS to check for a cycle in a directed, can we simply check for the algorithm visiting a marked node? Explain yes or no. [10 points]

8. In the DFS based topological sort, what would happen if nodes were pushed on the stack before their children were explored, rather than after? Would the algorithm still work? Explain. [10 points]

Programming: Binary Search Trees

9. One of the data structures we studied is Binary Search Trees (BST). A simple implementation of node for a binary tree is shown below. For this question, you are to implement a method called isBST that takes first node in a binary tree (its root) and returns true if it is the root of a BST, and false otherwise. Be careful with your syntax. [30 points]

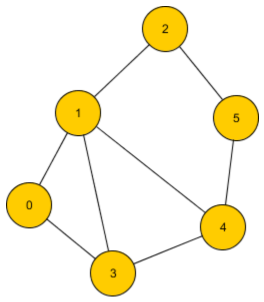
```
private class Node {
    public final Key key;
    public Value val;
    public Node left, right;
    public int N;

    public Node(Key key, Value val, int N) {
        this.key = key; this.val = val; this.N = N;
    }
}


private static boolean isBST(Node root) {
```

Programming: Hash Tables

FYI: if this was the real exam, you would probably be looking at a question asking you to write a hashfunction. Or something.

Programming: Undirected Graphs

10. Run the **BFS** algorithm on this graph to compute the shortest paths between 0 and every other node. For reference, the BFS algorithm is shown on the next page. Use the adjacency list above for the order of the nodes explored and follow the trace format shown before. Your answer must include the values of *v*, *queue*, and *edgeTo*, as they update. [20 points]



0	3, 1
1	0, 4, 3, 2
2	1, 5
3	1, 0, 4
4	1, 3, 5
5	2, 4

Figure 1: Sample graph.

Before loop:
marked[ ]=
queue={ }

Loop 1:
v=
edgeTo[ ]=
edgeTo[ ]=
queue = { }

Algorithm 1 BFS for pathing.

```
public class BreadthFirstPaths {
    private boolean[] marked;
    private int[] edgeTo;
    private final int s;
    public BreadthFirstPaths(Graph G, int s) {
        marked = new boolean[G.V()];
        edgeTo = new int[G.V()];
        this.s = s;
        bfs(G, s);
    }

    private void bfs(Graph G, int s) {
        Queue<Integer> queue = new LinkedList<>();

        marked[s] = true;
        queue.add(s);

        while (!queue.isEmpty()) {
            int v = queue.remove();
            for (int w : G.adj(v))
                if (!marked[w]) {
                    edgeTo[w] = v;
                    marked[w] = true;
                    queue.add(w);
                }
        }
    }

    public boolean hasPathTo(int v) {
        return marked[v];
    }

    public Iterable<Integer> pathTo(int v) {
        if(!hasPathTo(v))
            return null;
        Stack<Integer> path = new Stack<>();
        for(int x = v; x != s; x = edgeTo[x])
            path.push(x);
        path.push(s);
        return path;
    }
}
```