Com S 228
Fall 2014
Exam 3: Final Exam

DO NOT OPEN THIS EXAM UNTIL INSTRUCTED TO DO SO

Name: _____

ISU NetID (username): _____

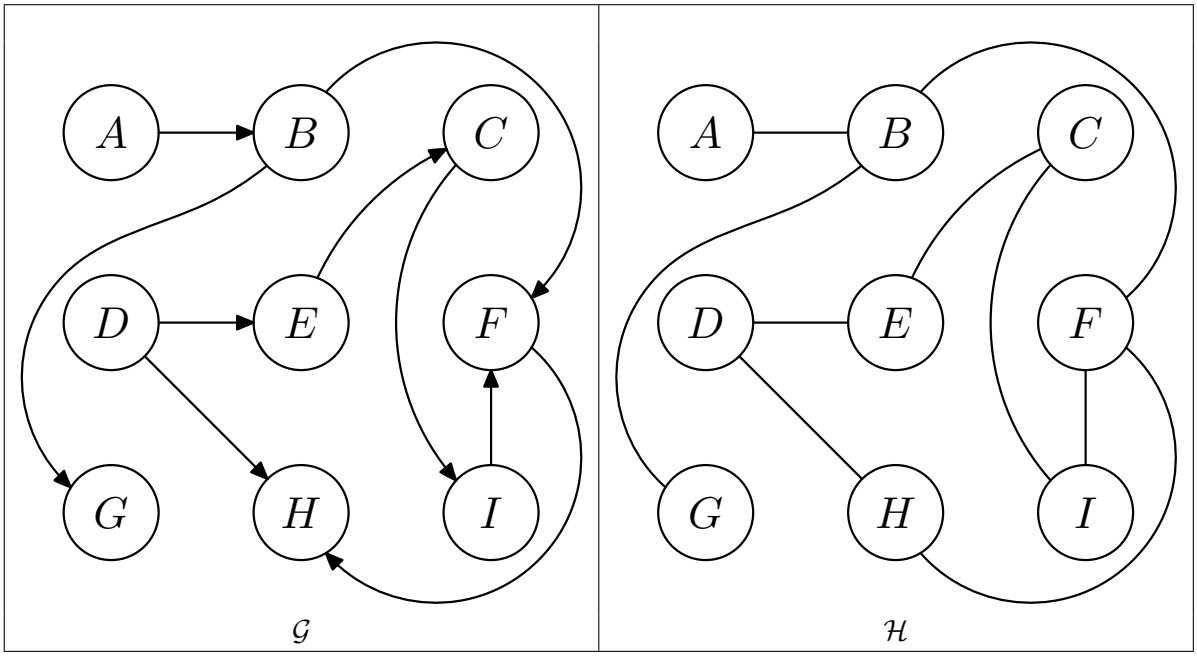
Closed book/notes, no electronic devices, no headphones. Time limit 105 minutes. Partial credit may be given for partially correct solutions.

- Use correct Java syntax for writing code.
- You are not required to write comments for your code; however, brief comments may help make your intention clear in case your code is incorrect.

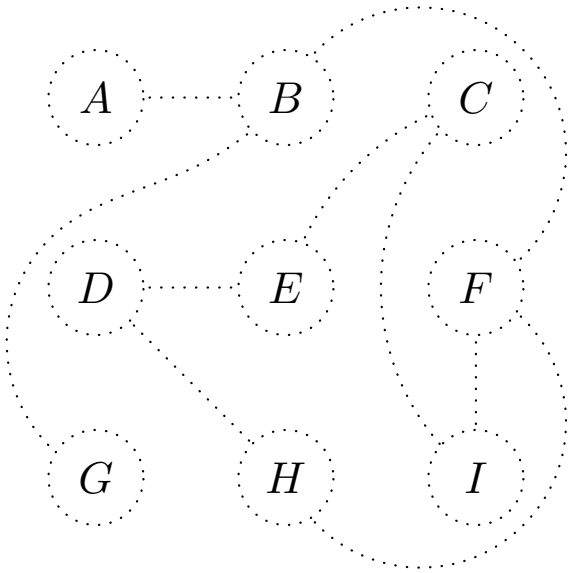
If you have questions, please ask!

Question	Points	Your Score
1	18	
2	12	
3	12	
4	10	
5	30	
6	18	
Total	100	

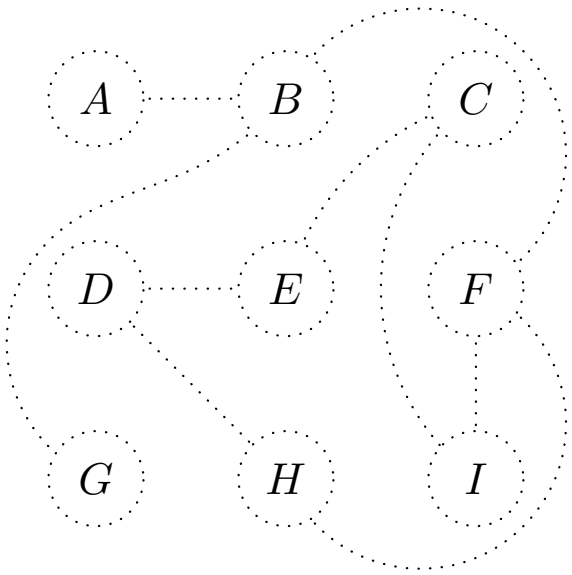
1. (18 pts) The parts of this problem refer to the two graphs, $\mathcal{G}$ and $\mathcal{H}$, below. In all parts, when you must choose between multiple nodes, make the decision based on the alphabetical order of the vertex names (choose a node whose name is earlier in the alphabet before one whose name is later, which means that, unless explicitly instructed otherwise, you start at A). Darken the edges and nodes (circles) in the pre-drawn templates according to the instructions given for each part. Nodes and edges should only be darkened if they are in the answer, e.g., if you are asked for the shortest path from A to B, you will darken the circles around A and B and the edge that connects them.



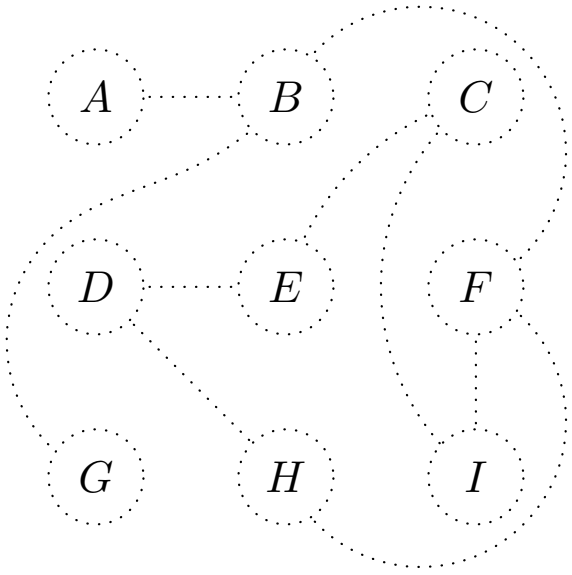
(a) (6 pts) Give the DFS forest for the directed graph $\mathcal{G}$.



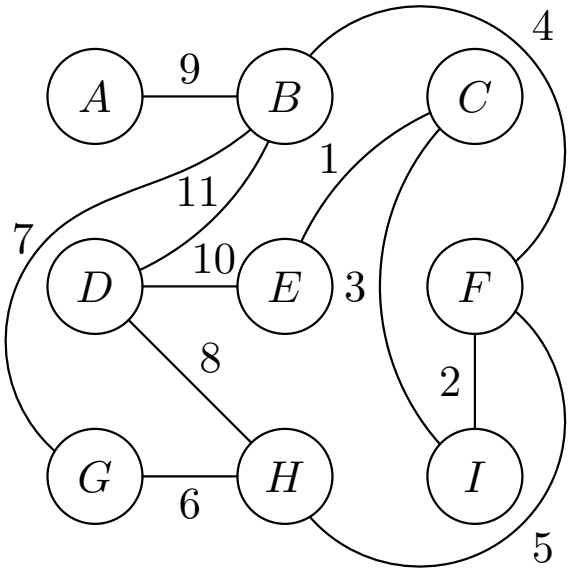
(b) (6 pts) Give the BFS tree for the directed graph $\mathcal{G}$. Mark all nodes, including unreachable nodes, with their distance from A.



(c) (6 pts) Give the BFS tree for the undirected graph $\mathcal{H}$ starting with node D. Mark all nodes, including unreachable nodes, with their distance from D.

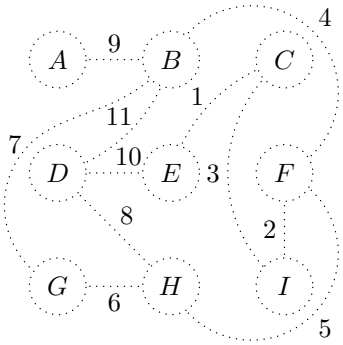


2. (12 pts) The parts of this problem refer to the weighted graph, $\mathcal{G}$, below. Darken nodes and edges as prescribed in the previous problem.

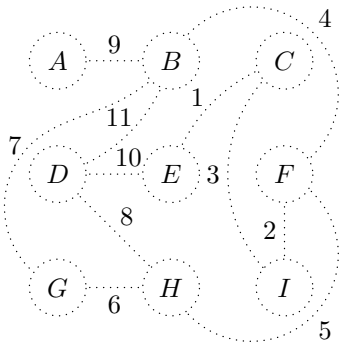


$\mathcal{G}$

(a) (6 pts) Give a minimum spanning tree of $\mathcal{G}$.

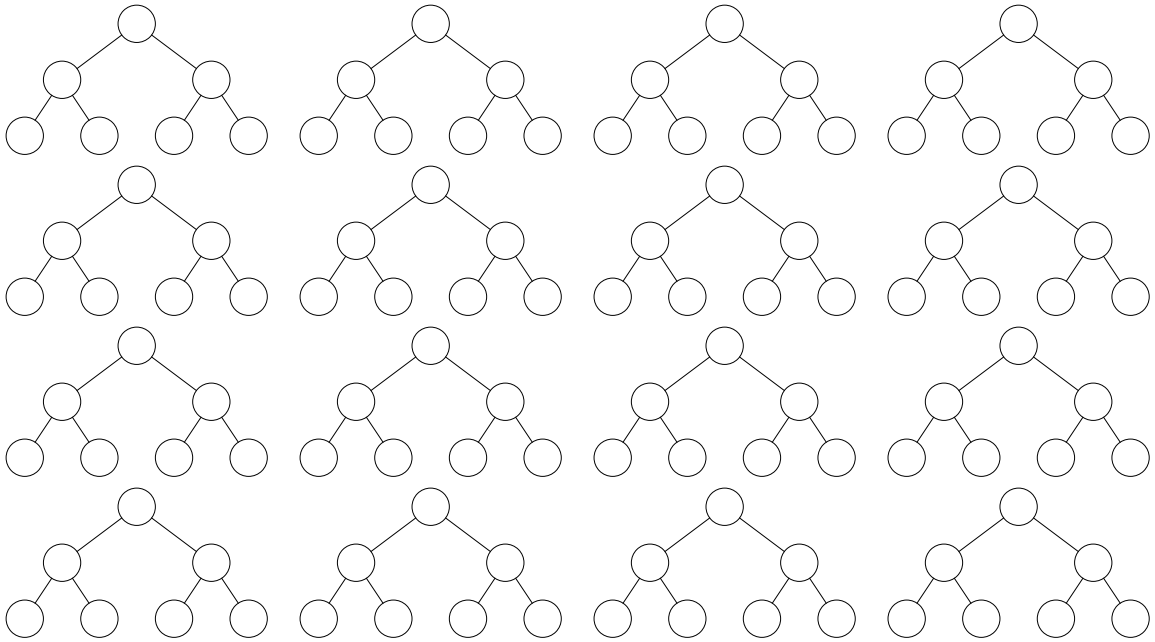


(b) (6 pts) Give a shortest path in $\mathcal{G}$ from G to E.



3. (12 pts) The following table illustrates the operation of HEAPSORT on the array in row 0. Every subsequent row is calculated by performing a single swap as determined by HEAPSORT on the preceding line. Fill in the missing lines and underline the first line that is a heap (in other words, demarcate the transition from HEAPIFY to the sort proper). You may use the empty trees below for scratch. **Only the table is used for grading! The trees are just tools for you and will not be evaluated.**

Row	Array						
0	0	4	1	5	3	2	6
1	0	4	6	5	3	2	1
2							
3	6	5	0	4	3	2	1
4							
5	1	5	2	4	3	0	6
6							
7	5	4	2	1	3	0	6
8							
9	4	0	2	1	3	5	6
10							
11	0	3	2	1	4	5	6
12	3	0	2	1	4	5	6
13	3	1	2	0	4	5	6
14							
15	2	1	0	3	4	5	6
16	0	1	2	3	4	5	6
17							
18	0	1	2	3	4	5	6

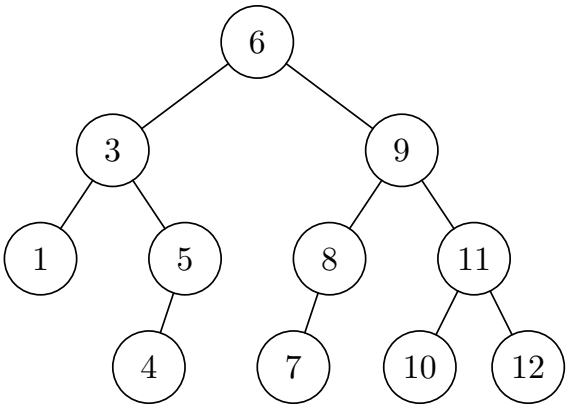


4. (10 pts) The following problems relate to binary search trees.

   (a) (5 pts) Draw the binary search tree that results from inserting the values in the following array, from left to right, into an empty, unbalanced BST.

4	2	6	3	5	7	1

   (b) (5 pts) Draw the BST that results from the removal of 9 from the following unbalanced BST where replacement uses the successor.



5. (30 pts) You have access to a queue class, `Queue<E>`, with methods `void enqueue(E)`, `void dequeue()`, and `E front()`. `enqueue()` adds an item to the end of the queue. `dequeue()` removes the front-most item from the queue, or does nothing with an empty queue. `front()` returns the value stored in the front-most item of the queue, or `null` if the queue is empty. All the defined methods are fully implemented, work as specified, and always succeed.

   You are working inside a binary tree class. **This is not a binary search tree!** The nodes of the tree are members of the internal class `Node`, defined as follows:

```
private class Node {
    public Node left, right, parent;
    public Integer value;
}
```

   and used in the canonical way, i.e.: all nodes, save the root, have a non-null parent, referenced by `parent`, and are either the left or right child of that parent, all internal nodes have at least one child, referenced by `left` or `right`, whose parent is that node, and every node stores an integer value in `value`.

   Your binary tree is correctly formed. It has a root node, `Node root`. A null root implies an empty tree. **You may assume nothing further about the binary tree, node, or queue, nor may you use any other APIs.** Implement the following methods without using any helper methods. Throw a `NullPointerException` if the client calls either method on an empty tree.

(a) (15 pts) `max()` **recursively** finds and returns the largest item in the subtree rooted at `current`.

```
public Integer max(Node current)
{
  // Not a binary search tree!  Just a general binary tree.

  // It's a good thing I didn't implement this as if for a
  // binary search tree, else I wouldn't have done well!
}
```

(b) (15 pts) `min()` **iteratively** finds and returns the smallest item in the tree. *Hint: remember level-order traversals!*

```
public Integer min()
{
  // Not a binary search tree!  Just a general binary tree.

  // It's a good thing I didn't implement this as if for a
  // binary search tree, else I wouldn't have done well!
}
```

6. (18 pts) Give the tightest-possible asymptotic complexity of the following operations and values. **Use correct notation!**

   (a) (2 pts) Removing the smallest item from a binary min-heap with n items.

   (b) (2 pts) Inserting an item into a hash set with n items.

   (c) (2 pts) Heap sort on an n-element array.

   (d) (2 pts) Traversing every edge in a connected, undirected graph with V vertices and E edges, built with adjacency lists.

   (e) (2 pts) The minimum height of a binary tree containing n nodes.

   (f) (2 pts) The maximum height of a binary tree containing n nodes.

   (g) (2 pts) The maximum number of vertices in a connected graph with E edges and no self loops.

   (h) (2 pts) The number of edges in a fully connected graph with V vertices.

   (i) (2 pts) Your score on this exam out of $n = 100$ points.

Scratch paper

Scratch paper

Scratch paper

Scratch paper