Final Exam for Com S 228 Spring 2017

Name:

University ID:

There are 10 pages and 5 questions. Please start with easy questions. We will give partial credit for a partially correct solution. You are not allowed to use any books, notes, calculators, or electronic devices. The exam is 120 minutes long.

Question	Points	Your Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

1 (20). For each of the methods below, give a tight big-O bound for the worst-case running time of the method as a function of n. Use formal notation. **Use correct notation!**

(a) Performing `find(item)` operation on a binary search tree with n elements.

(b) Performing a pre-order traversal on a binary tree with n elements.

(c) Removing the largest element from a max heap with n elements.

(d) Performing Heap Sort on an array with n elements.

(e) Performing a breadth-first search on a graph with n vertices and $6n$ edges.

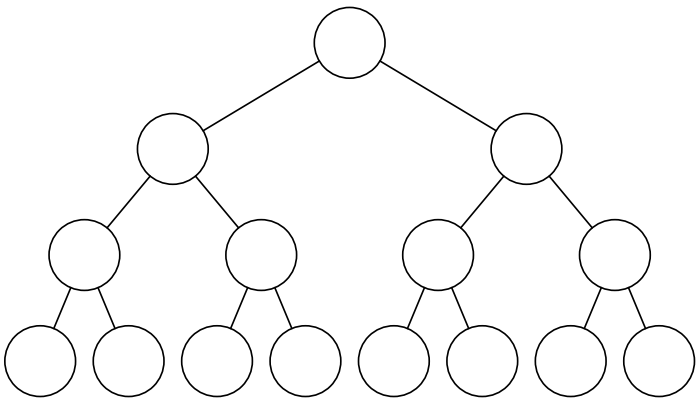
(f) Running Dijkstra's algorithm on a directed graph with n vertices and $2n+3$ edges.

2 (20). Recall that an array is used to implement a max heap as a complete binary tree in which the left and right subtrees of any node contain only elements smaller than or equal to the node's element. Let n be the number of elements in the input array. For this exercise you will show the result of applying these methods:

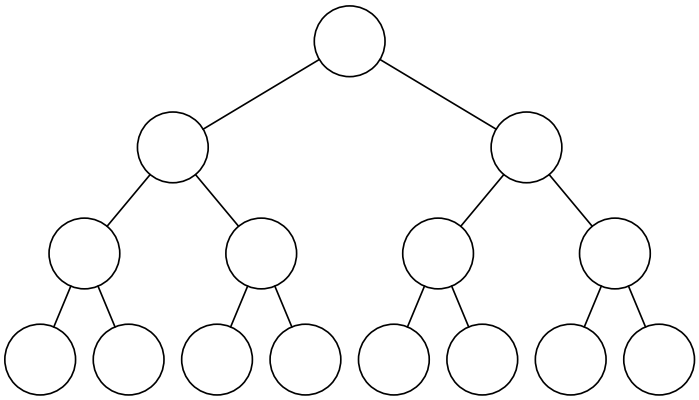
- `removeMax()` returns the largest element in the max heap. Before this method returns, it replaces the value at index 0 with the value at index $n-1$ and calls `percolateDown(0)` to rearrange the elements $A[0]...A[n-2]$ such that the array A still represents a max heap in $O(\log n)$. The number of elements in the array is reduced by one after this method returns.
- `add(E element)` adds the element in the array A and calls `percolateUp()` to rearrange the elements in the array in $O(\log n)$ such that the max heap property is maintained. The number of elements in the array is increased by one after this method returns.

(a) Use the binary tree template to illustrate the max heap corresponding to the values of the elements stored in the array A when A has the following elements from index 0 to index 11. Suppose elements are of type Integer. The template may contain more nodes than are necessary to complete the heap. Leave a node empty to signify that it is not part of the heap. Please do your work outside of the template; write only your complete answer within it.

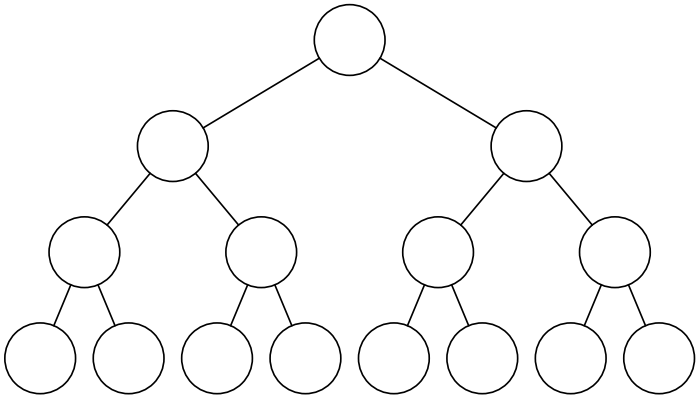
`A: 65 55 50 40 25 30 15 20 5 10 14 17`



(b) Fill in the max heap after `removeMax()` is called once on the heap. Show only the final result.



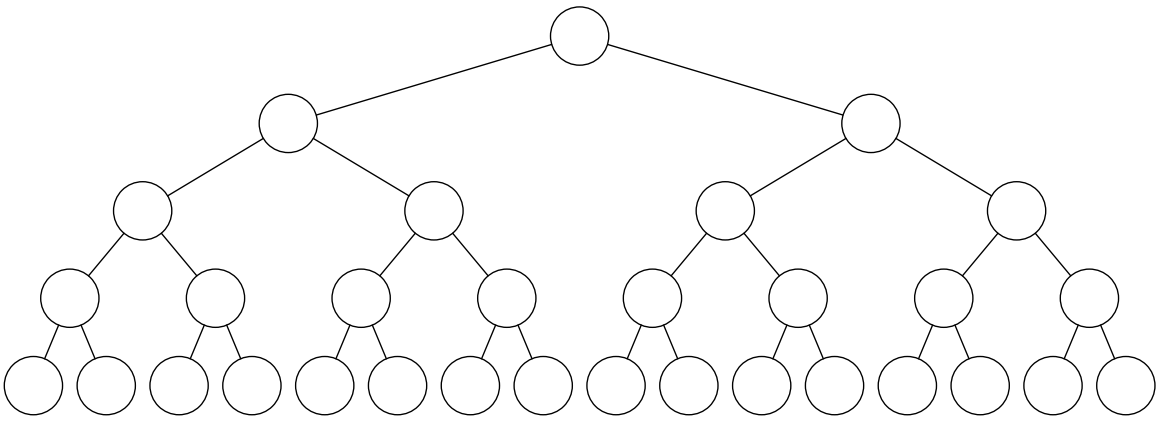
(c) Fill in the max heap after `add(35)` is called on the resulting heap from the previous step. Show only the final result.



3 (20). Let `CS228BinarySearchTree<E extends Comparable<? super E>>` be an unbalanced binary search tree that adds smaller items to the left. A binary search tree is constructed by the code below.

```
CS228BinarySearchTree<Integer> bstree = new CS228BinarySearchTree<Integer>();
bstree.add(35);
bstree.add(40);
bstree.add(10);
bstree.add(50);
bstree.add(60);
bstree.add(0);
bstree.add(25);
bstree.add(15);
bstree.add(20);
bstree.add(45);
bstree.add(5);
bstree.add(30);
```

(a) Draw the binary search tree constructed by the above code in the template below. As in the preceding problem, only write your final answer in the template and leave nodes empty to signify that they are not part of the tree.

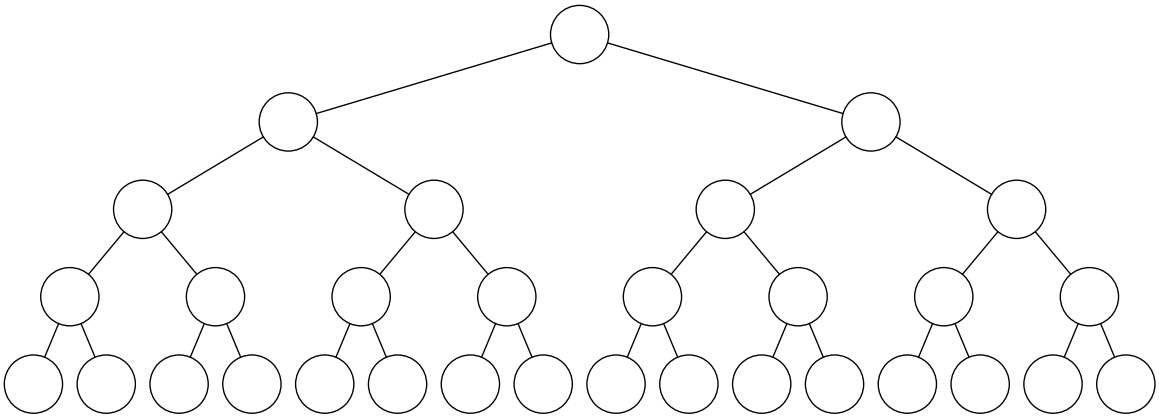


(b) Show an ordering of integer values in a pre-order traversal.

(c) Show an ordering of integer values in an in-order traversal.

(d) Show an ordering of integer values in a post-order traversal.

(e) Perform the statement `bstree.delete(10)` (based on the approach used in lecture) and fill in the resulting tree.



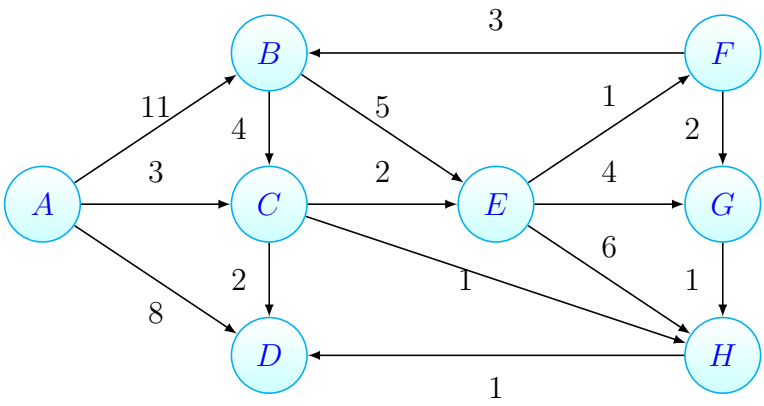
4 (20). Write a public iterative method named `removeMax()` for the binary search tree class below. The method removes the node with the largest `data` item and returns the largest `data` item if the tree is not empty. Otherwise, it returns `null`. Note that for any node in the tree, the `data` items of its left subtree must be smaller than the `data` item of the node, and the `data` items of its right subtree must be greater than the `data` item of the node. You are not allowed to use any known method for the binary search tree class. Note again that the `removeMax()` method has to be iterative. After the method is completed, the resulting tree has to be a binary search tree.

```
public class CS228BST<E extends Comparable<? super E>>
{ private Node root;
  private class Node
    { public Node left;
      public Node right;
      public Node parent;
      public E data;

      public Node(E item)
      { data = item;
        left = right = parent = null;
      }
    }
```

Complete the code template on next page.

```
  public E removeMax()
  {
    if ( root == null )
    {
                                                          // <- ANSWER
    }

    Node cur = root;
    while (                              )                // <- ANSWER
    {
                                                          // <- ANSWER
    }

    E data = cur.data;
    if ( cur.parent != null )
    {
                                                          // <- ANSWER

    }
    else
    {
                                                          // <- ANSWER

    }

    if ( cur.left != null )
    {
                                                          // <- ANSWER

    }

    return            ;                                   // <- ANSWER
  }
}
```

5 (20). Answer the following questions on the directed graph below.



(a) Show an ordering of vertices in a breadth-first traversal with the vertex "A" as the source, assuming that you visit the neighbors of the current vertex in the alphabetical order. For example, a search at the vertex "B" visits its neighbors "C" and "E" in this order.

(b) Show an ordering of vertices in a depth-first traversal with the vertex "A" as the source, again assuming that you visit the neighbors of the current vertex in the alphabetical order.

Continue on next page.

(c) Show the distance and predecessor values for each vertex at the end of Dijkstra's algorithm with the vertex "A" as the source.

Vertex	Distance	Predecessor

(d) Show a shortest path from the vertex "A" to the vertex "D".