Com S 228

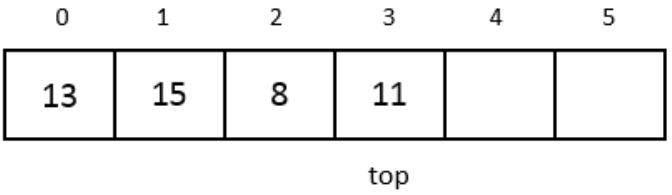
Spring 2014

Final Exam Sample Solution

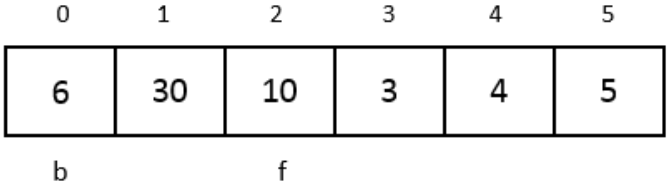
1a) $O(1)$
b) $O(\log n)$
c) $O(\log n)$
d) $O(n)$
e) $O(n)$
f) $O(n)$
g) $O(n+m)$

2a) `a b 2 c d + ^ - 2 ^ ^ a b c 4 5 c d + / - / - b * * -`
b) `(a + c * (a + c * (a + b))) – ((a % b) % c) % d`
c) $O(n)$; $O(n)$

3a)



b)



c)

```
public static <E> void reverseStack(Stack<E> stk>
{
    if ( stk == null || stk.isEmpty() ) return;
    LinkedQueue<E> tmp = new LinkedQueue<E>();

    //TODO

    LinkedQueue<E> q = new LinkedQueue<E>();
    while (!stk.isEmpty())
```

```
    {
        q.enQueue(stk.pop());
    }

    while (!q.isEmpty())
    {
        stk.push(q.deQueue());
    }
}
```

4a)

Pre-order: 15, 5, 3, 12, 10, 6, 7, 13, 16, 20, 18, 23

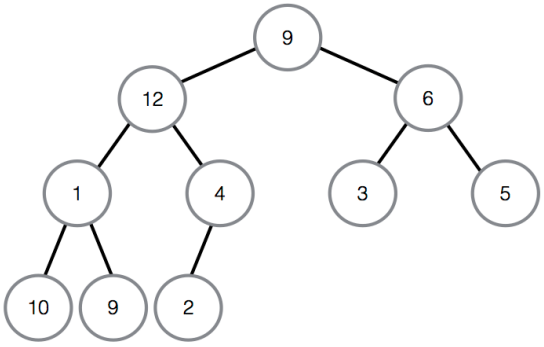
In-order: 3, 5, 6, 7, 10, 12, 13, 15, 16, 18, 20, 23

Post-order: 3, 7, 6, 10, 13, 12, 5, 18, 23, 20, 16, 15

b)

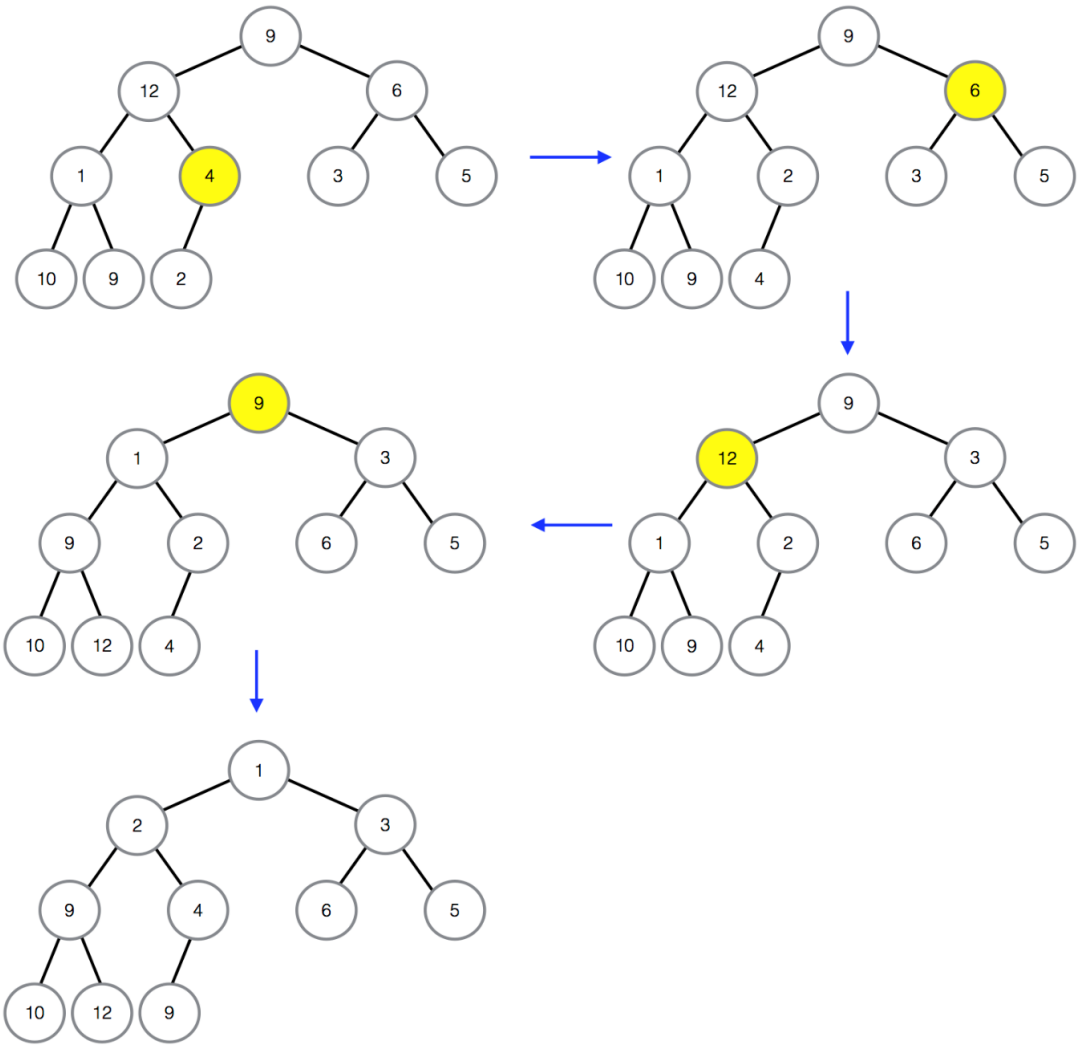
```
private String levelOrderTraversal()
{
    //TODO

    String str = "";

    if (root == null) return str;

    LinkedQueue<E> q = new LinkedQueue<E>();
    q.enqueue(root);

    while (!q.isEmpty())
    {
        Node curNode = q.dequeue();
        str = str + " " + curNode.data.toString();
        Node child = curNode.firstChild;

        while (child != null)
        {
            q.enqueue(child);
            child = child.nextSibling;
        }
    }

    return str;
}
```

5a) Complete binary tree



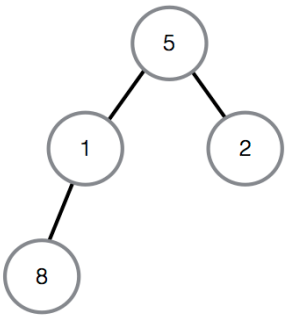
b) Min heap (full credit if correct final answer)

1	2	3	9	4	6	5	10	12	9



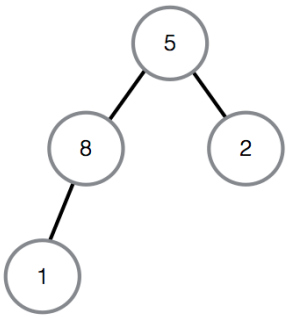
c) Required intermediate and final results from heapsort:

1)



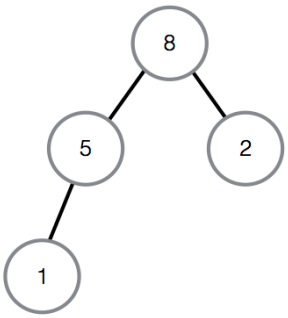
5	1	2	8

2)



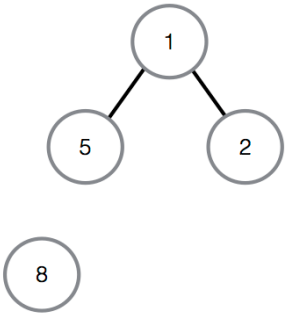
5	8	2	1

3)



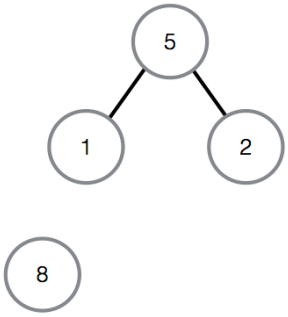
8	5	2	1

4)



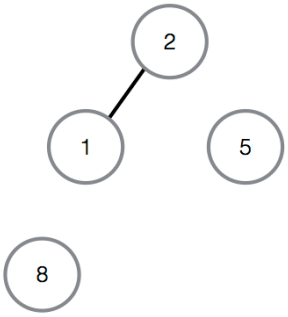
1	5	2	8

5)



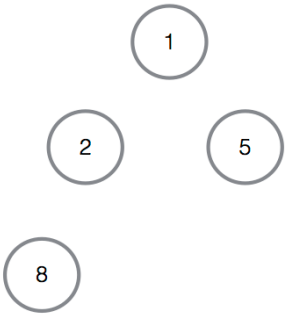
5	1	2	8

6)



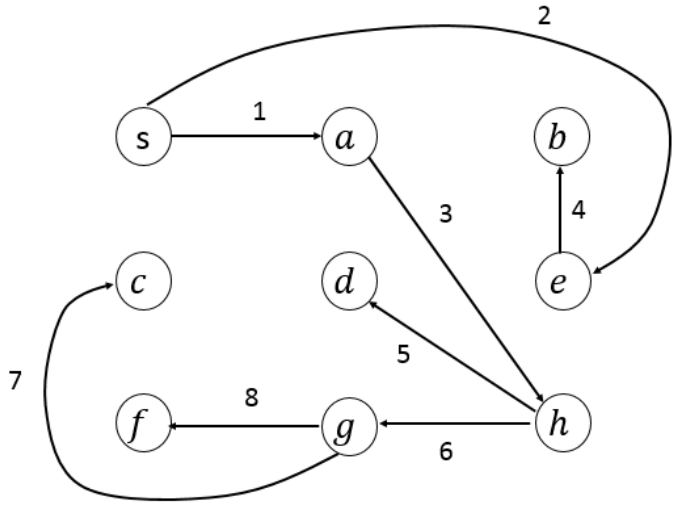
2	1	5	8

7)



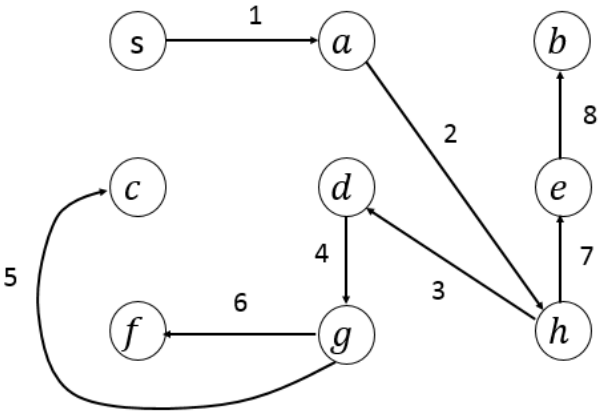
1	2	5	8

a) BFS tree



vertex V	s	a	b	c	d	e	f	g	h
dist(V)	0	1	2	4	3	1	4	3	2

b) DFS tree



c) Yes; Yes

7.

```
public E removeRoot() throws IllegalStateException
{
      E data;

      if (size == 0)
            throw new IllegalStateException("No root removal on an empty tree");

      data = root.data;


      // BST has only one node. (1 pt)
      if (size == 1)
      {
          root = null;
      }

      // BST has two or more nodes but no left subtree. (2 pts)
      else if (root.left == null)
      {
            root = root.right;   // root must have a right child since size > 1.
            root.parent = null;
      }

      // BST has two or more nodes and a left subtree.
      // find the predecessor of the root and promote it to the new root.
      else
      {
            Node cur = root.left;
            Node prnt = root;

            // find the predecessor of the root.
            while (cur.right != null)
            {
                  prnt = cur;
                  cur = cur.right;
            }

            // left child of root has a right subtree.
            if (prnt != root)
            {
                  // update links related to the predecessor's subtree(s).
                  // (3 pts)
                  prnt.right = cur.left;
                  if (cur.left != null)
                        cur.left.parent = prnt;

                  // set up the new root's left subtree. (2 pts)
                  cur.left = root.left;
                  cur.left.parent = cur;
            }
```

```
            // set up the new root's right subtree. (3 pts)
            cur.right = root.right;
            if (cur.right != null)
                  cur.right.parent = cur;


            // set up the new root. (2 pts)
            cur.parent = null;
            root = cur;
      }

      // other updates if needed (1 pt)
      size--;
      return data;
}
```