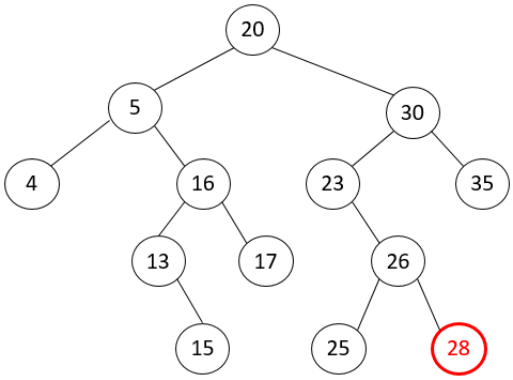
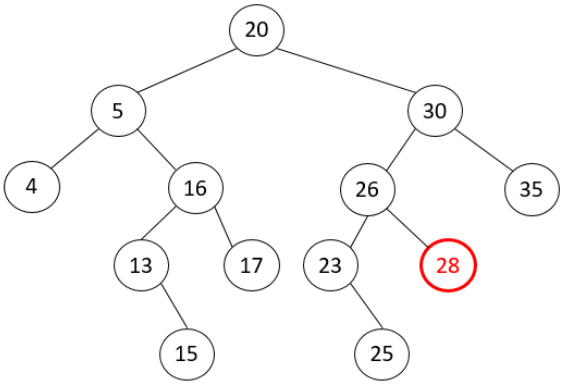
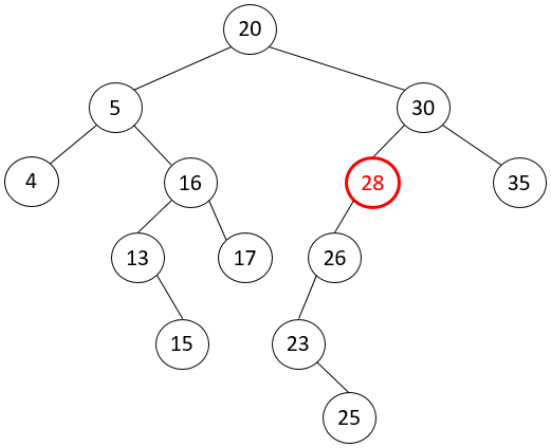
Com S 228
Fall 2015
Final Exam Sample Solution

1a) 4
b) 2
c) 3
d) 4, 15, 13, 17, 16, 5, 25, 26, 23, 35, 30, 20
e) BST insert:

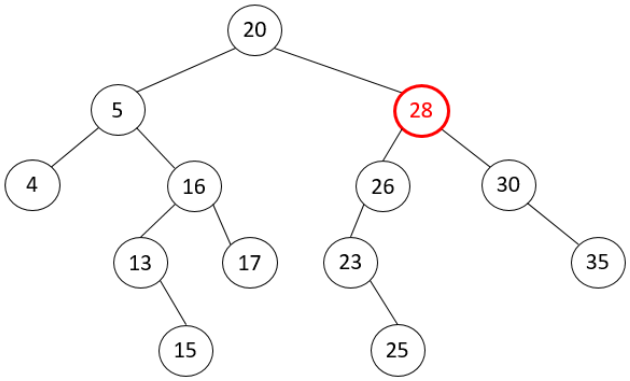


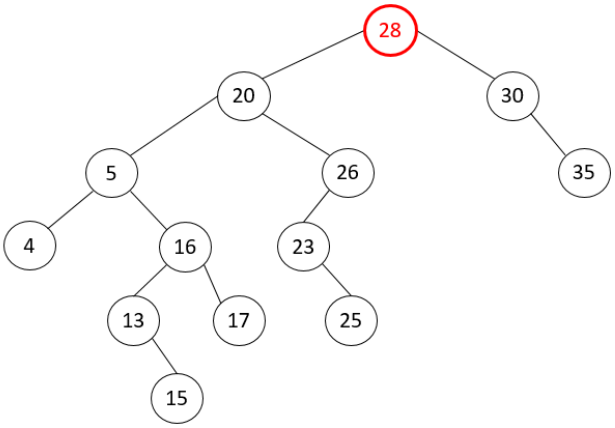
Zig-zig:



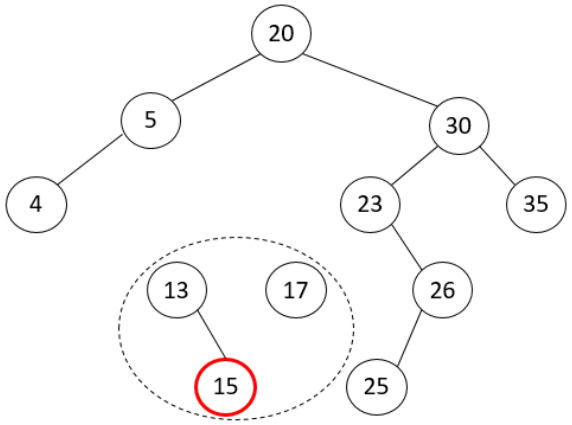


Zig-zag:

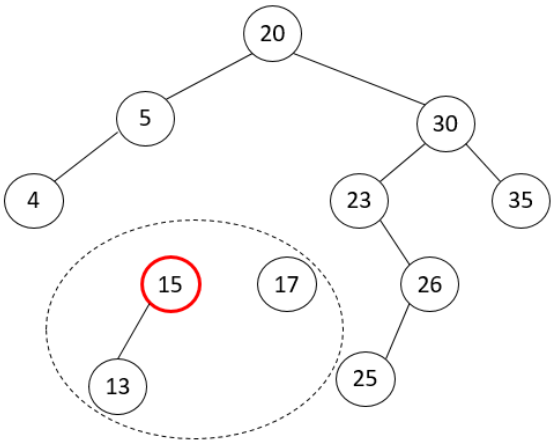


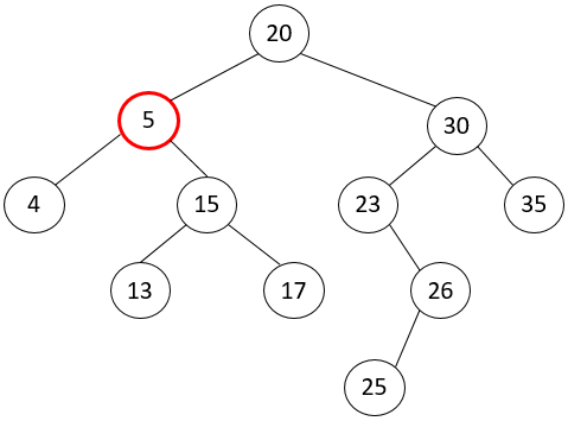


f) Remove 16:

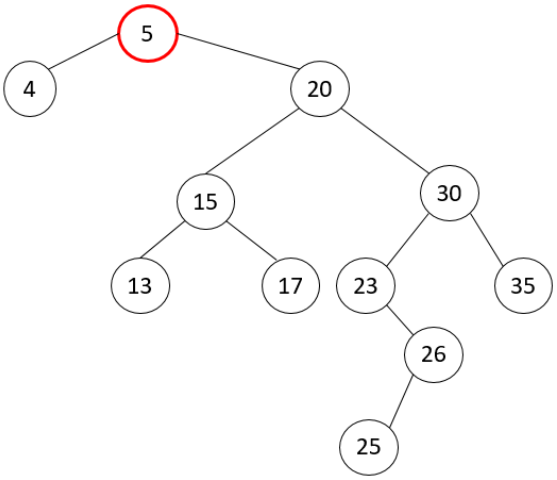


Join the subtrees of 15:





Splay at the (former) parent 5 (zig):



2a)

Key	14	12	23	11	35	40	5
Hash code	0	9	9	8	10	4	2

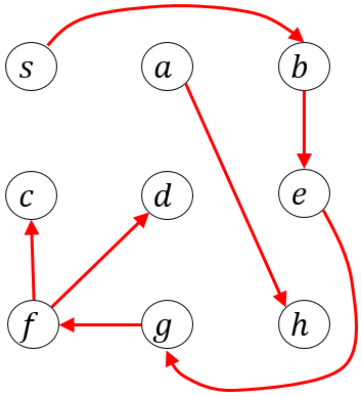
b)

14	35	5		40				11	12	23
0	1	2	3	4	5	6	7	8	9	10

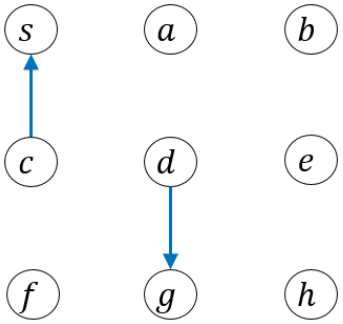
3. HEAPIFY ends on line 4.

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

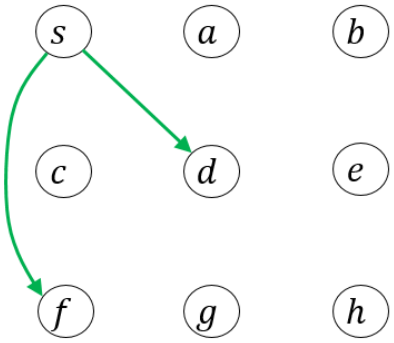
4a) 3
b) 2
c) DFS forest



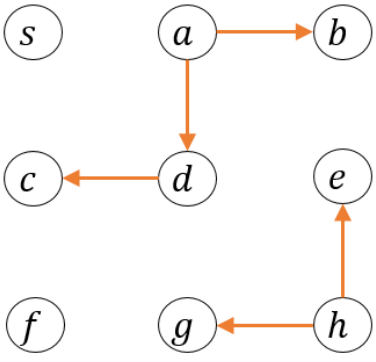
d) Back edges



e) Forward edges



f) Cross edges



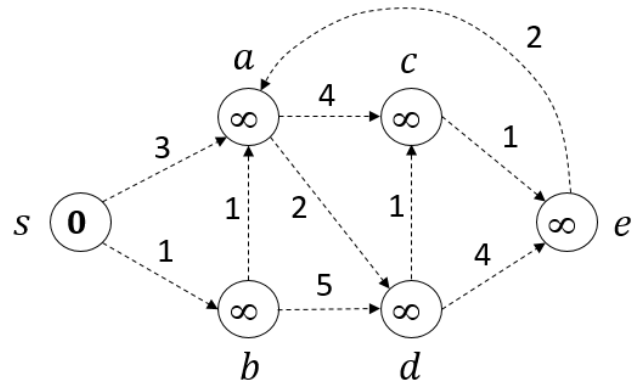
5a) Either of the following two answers.

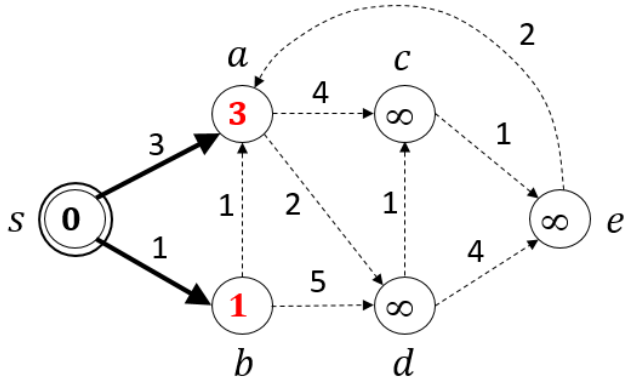
f	g	d	c	h	e	a	b

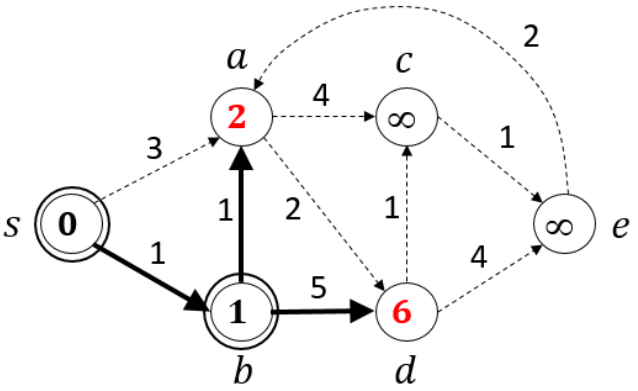
f	g	d	h	c	e	a	b

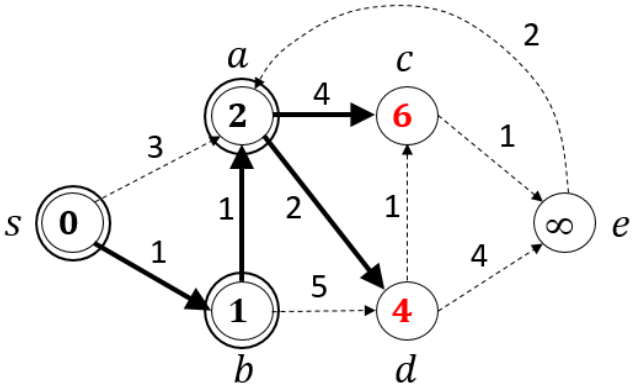
b) $O(n+m)$

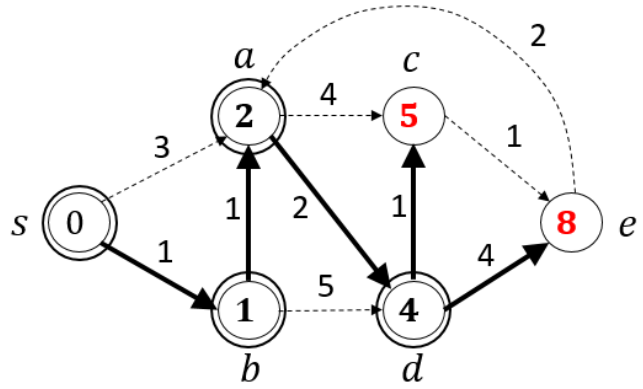
6.

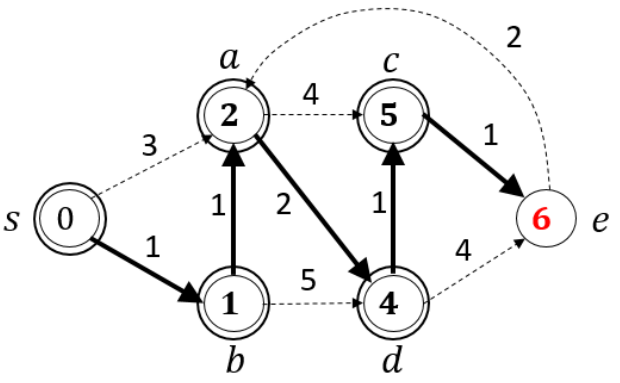




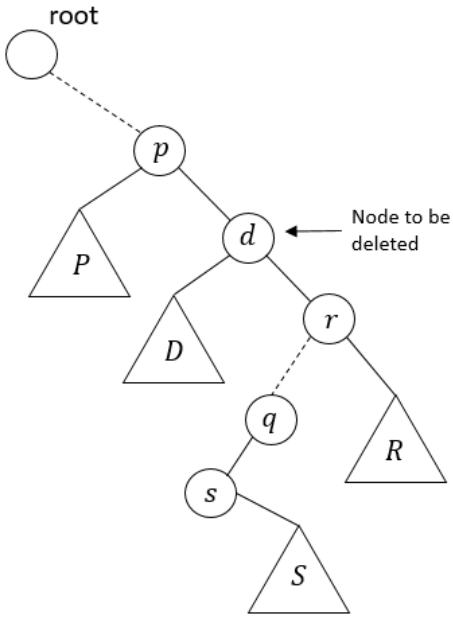








7a)




```
// make the subtree S a new subtree of q
// insert code below (2 pts)
q.left = s.right;
if (s.right != null)
     s.right.parent = q;

// make the left subtree of d the new left subtree of s
// insert code below (3 pts)
s.left = d.left;
if (d.left != null)
    d.left.parent = s;

// make the right subtree of d the new right subtree of s
// insert code below (3 pts)
s.right = r;      // or s.right = d.right
r.parent = s;     // or d.right.parent = s

// make the node p the new parent of s
// insert code below (4 pts)
s.parent = p;
if (p.left == d)
    p.left = s;
else
    p.right = s;
```

b)

```
/**
 * @return number of internal nodes having one child
 */
public int countOneChildNodes()
{
    // insert code below (1 pt)
    return countOneChildNodesRec(root);
}


/**
 * @param n
 * @return number of one-child parents in the subtree rooted at n
 */
private int countOneChildNodesRec(Node n)
{
    // n is null (base case for recursion)
    // insert code below (1 pt)
    if (n == null)
        return 0;


    // count the one-child parents in the left and right subtrees.
    // insert code below (4 pts)
    int lcount = countOneChildNodesRec(n.left);
    int rcount = countOneChildNodesRec(n.right);


    // return the count for the subtree rooted at n.
    // insert code below (6 pts)
    if ((n.left == null && n.right != null)
          || (n.left != null && n.right == null))
        return lcount + rcount + 1;
    else
        return lcount + rcount;
}
```