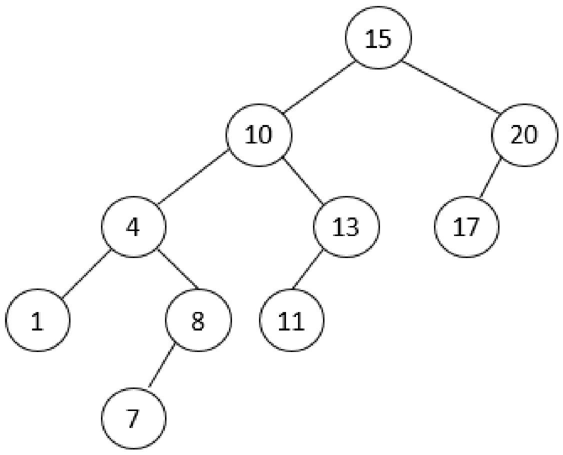
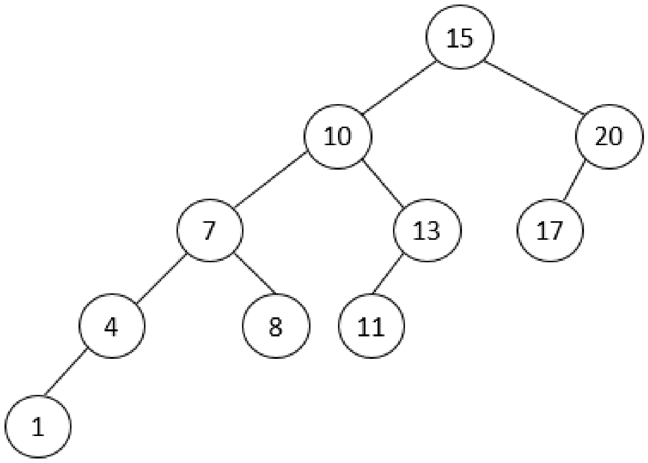
Com S 228

Spring 2015

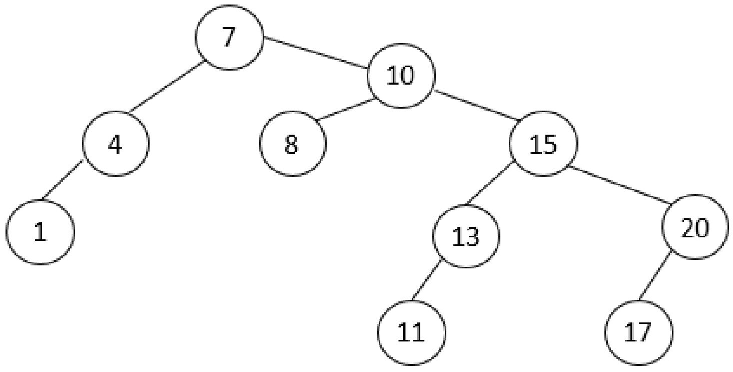
Final Exam Sample Solution

1a) Intermediate steps:

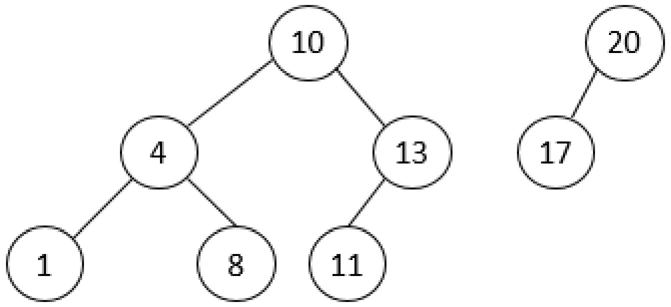


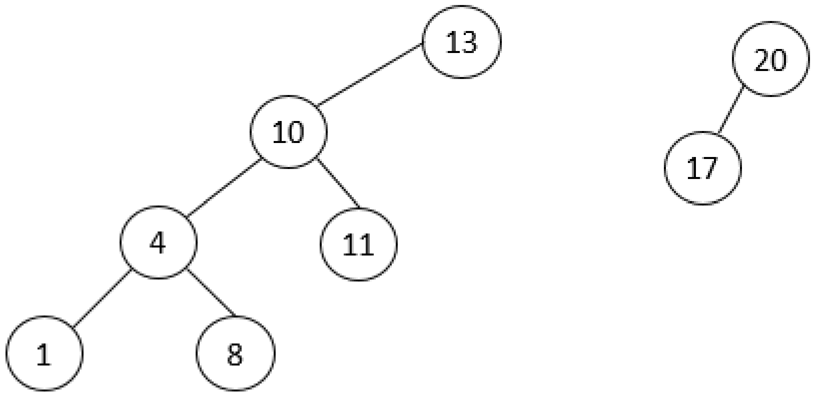


Answer:

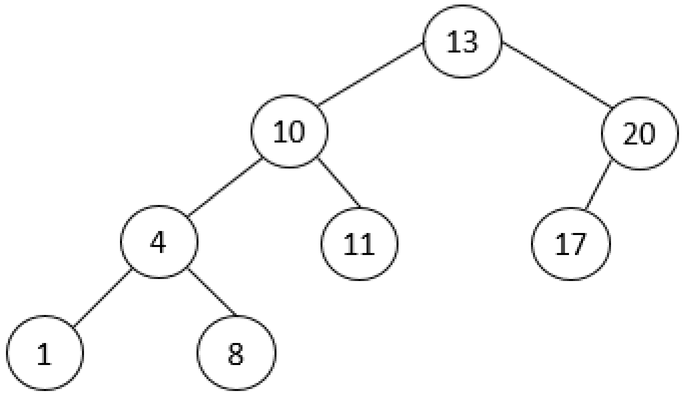


b) Intermediate steps:

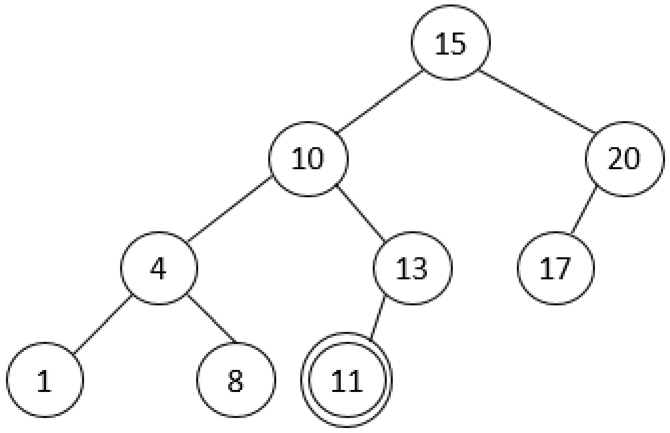


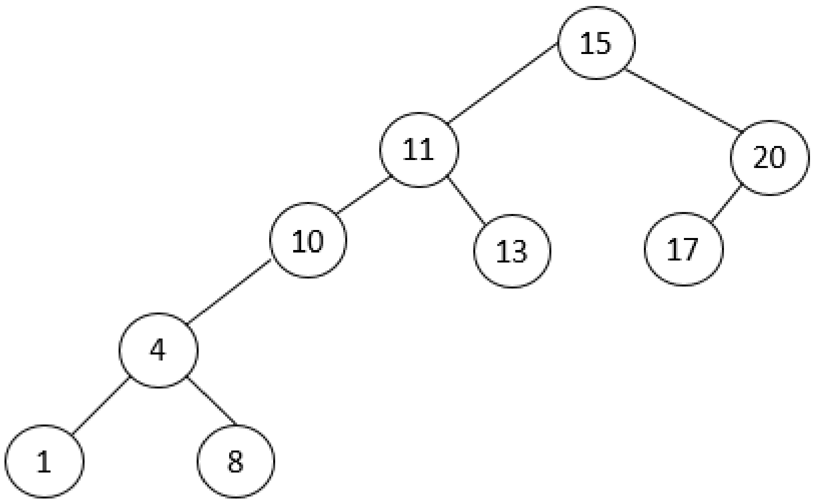


Answer:

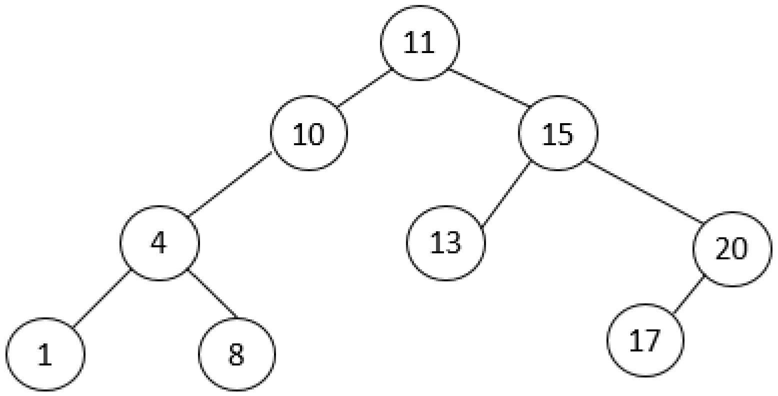


c) Intermediate steps:





Answer:



2. HEAPIFY ends on line 4.

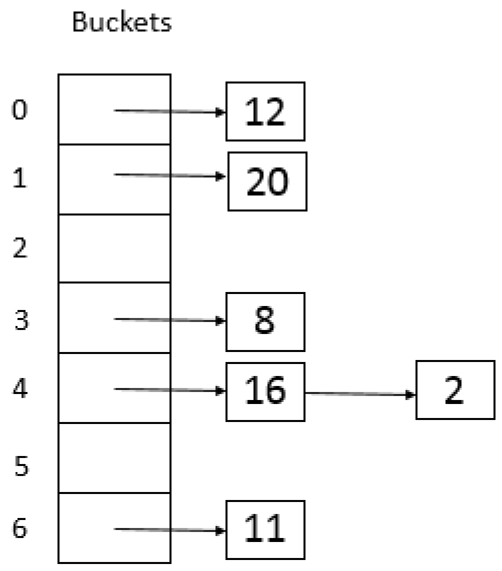
Row	Array						
0	1	3	0	4	6	2	5
1	1	3	5	4	6	2	0
2	1	6	5	4	3	2	0
3	6	1	5	4	3	2	0
4*	**6**	**4**	**5**	**1**	**3**	**2**	**0**
5	0	4	5	1	3	2	6
6	5	4	0	1	3	2	6
7*	**5**	**4**	**2**	**1**	**3**	**0**	**6**
8	0	4	2	1	3	5	6
9	4	0	2	1	3	5	6
10*	**4**	**3**	**2**	**1**	**0**	**5**	**6**
11	0	3	2	1	4	5	6
12	3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

3a)

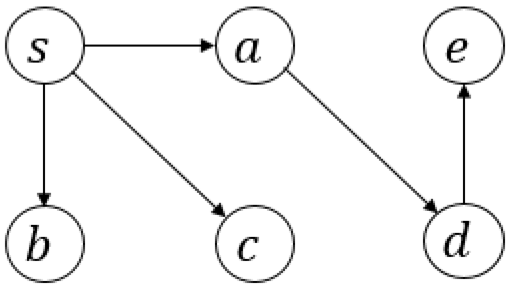
key	Hash code
11	6
20	1
2	4
8	3
12	0
16	4

b) It's not a perfect hash function.

c) The data structure representing a bucket is a singly linked list.

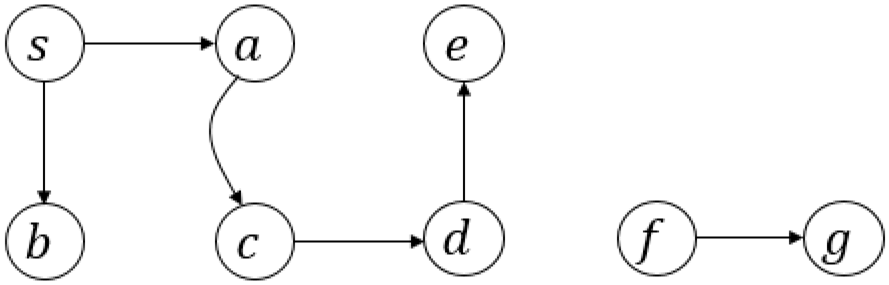


4a)

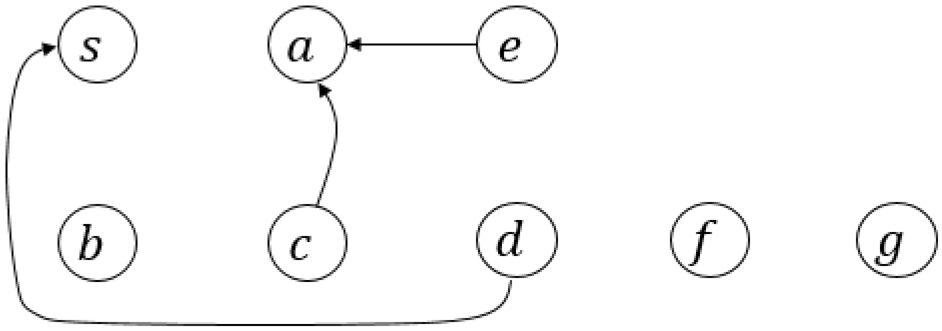


vertex v	s	a	b	c	d	e	f	g
pred(v)	null	s	s	s	a	d	null	null

b)



c)



d) No.

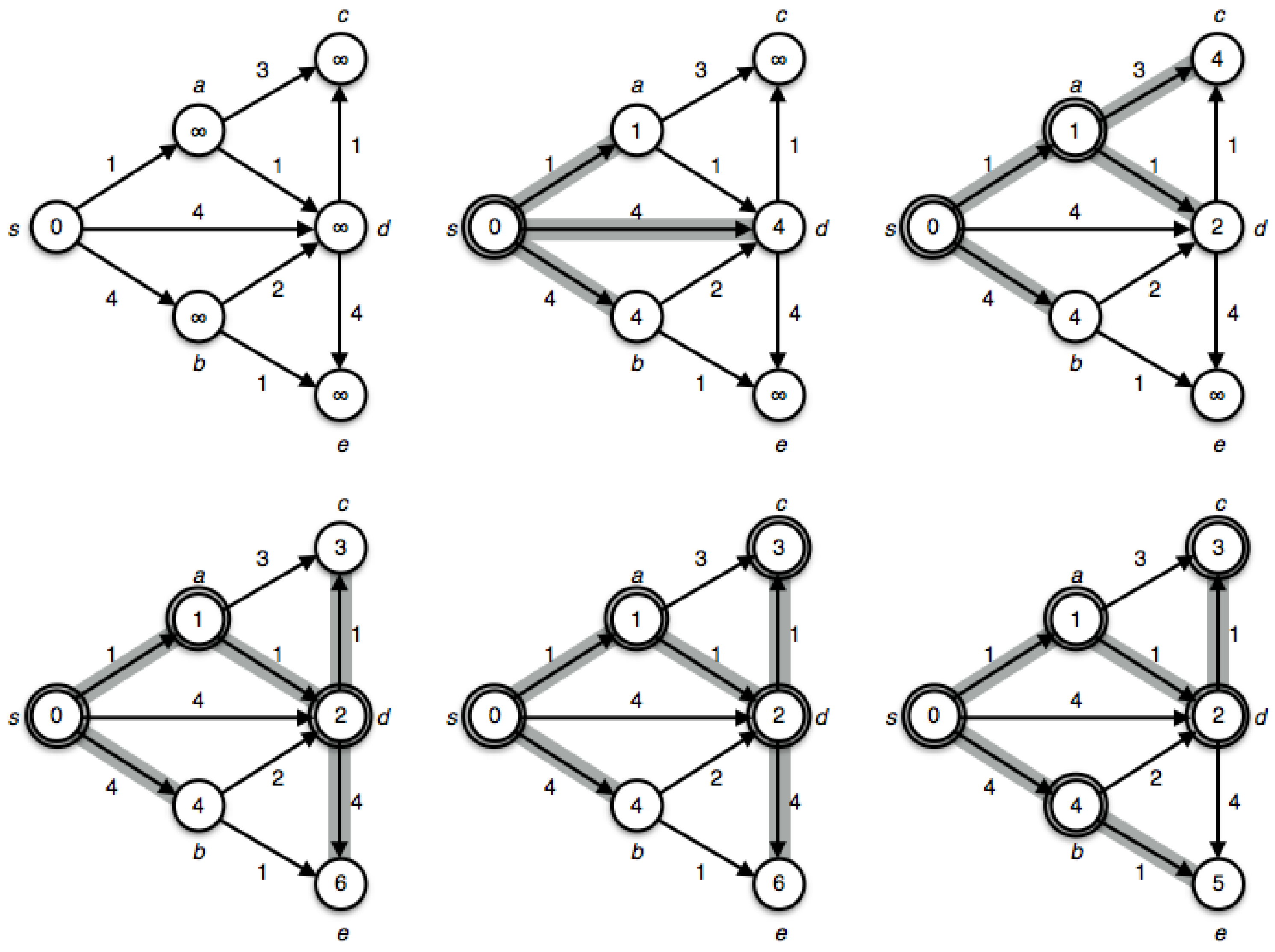
5. Either of the following three answers.

b	c	e	d	g	a	f

b	e	c	d	g	a	f

e	b	c	d	g	a	f

6.



7.

```
/**
 * Determine if the tree is equal to another tree (rooted at tree2),
 * that is, if the two trees are identical in structure and content.
 *
 * Precondition: tree2 != null
 *
 * @param tree2
 * @return  true if equal and false otherwise
 */
public boolean treeEqual(BST<E> tree2)
{
      // handle the situation where the two trees have different sizes.
      // inset code below (2 pts)
      if (size != tree2.size)
            return false;

      // the two trees have the same size.
      // inset code below (2 pts)
      return subtreeEqual(root, tree2.root);
}


/**
 * Recursively determine if two subtrees are equal.
 *
 * @param node1  root of the first subtree
 * @param node2  root of the second subtree
 * @return  true if equal and false otherwise
 */
private boolean subtreeEqual(Node node1, Node node2)
{
      // handle the situation(s) where one or both of the nodes are null.
      // inset code below (3 pts)
      if (node1 == null && node2 == null)
            return true;

      if (node1 == null && node2 != null)
            return false;

      if (node1 != null && node2 == null)
            return false;

      // neither node is null.
      // inset code below (4 pts)
      return node1.data.compareTo(node2.data)== 0
                  && subtreeEqual(node1.left, node2.left)
                  && subtreeEqual(node1.right, node2.right);
}
```

```
/**
 * Determine if the tree and another tree store the same set of keys.
 *
 * Precondition: tree2 != null
 *
 * @param tree2  tree to be compared with this tree
 * @return  true if set equal and false otherwise
 */
public boolean setEqual(BST<E> tree2)
{
      // handle the situation where the two trees have different sizes
      // inset code below (1 pt)
      if (size != tree2.size)
            return false;

      // the trees are of identical size. initialize two iterators
      // iter1 and iter2.
      // inset code below (2 pts)
      Iterator<E> iter1 = iterator();
      Iterator<E> iter2 = tree2.iterator();

      // compare whether the two sets of keys are equal.
      // inset code below (7 pts)
      while (iter1.hasNext())     // iter2.hasNext() == iter1.hasNext()
      {
            if (iter1.next().compareTo(iter2.next()) != 0)
                  return false;
      }

      return true;
}
```