PROLOGUE

FOR ALL ASSIGNMENTS

- **Attach a *prologue* for all assignments.**
- **Use sample *prologue* sheet in the course material, customize it for every assignment.**
- ***Prologue* makes it easy to separate assignments for grading purpose.**

EXERCISE 13

PROBLEM

Read the following numbers from an input file directly as integer values in to a binary tree. (Note: Do not assign an array with input values.).

55, 62, 89, 85, 97, 56, 71, 82, 38, 49, 25, 67, 58, 92, 100, 44, 69, 72, 65, 52, 41, 84, 21, 60, 95, 12, 35, 42, 105, 99, 34, 47, 35, 79, 95, 50, 25, 51

Using a binary tree structure read the numbers and form an inordered tree. All nodes to the left subtree of the root will be lower than root and all nodes in the right subtree are higher. Traverse the tree in preorder, inorder, postorder, and print the data value (info) in the node when the node is visited. Write a delete algorithm to delete nodes and replacing them with their inorder successor. Print the values inorder after all deletes. Write both non-recursive and recursive routines to traverse the trees.

User intends to find the following values in the tree. Verify and print message the whether the values are found in the tree or not:

71, 51, 38, 5, 0, 25, 42, 91, 35 and 47

DELIVERABLES

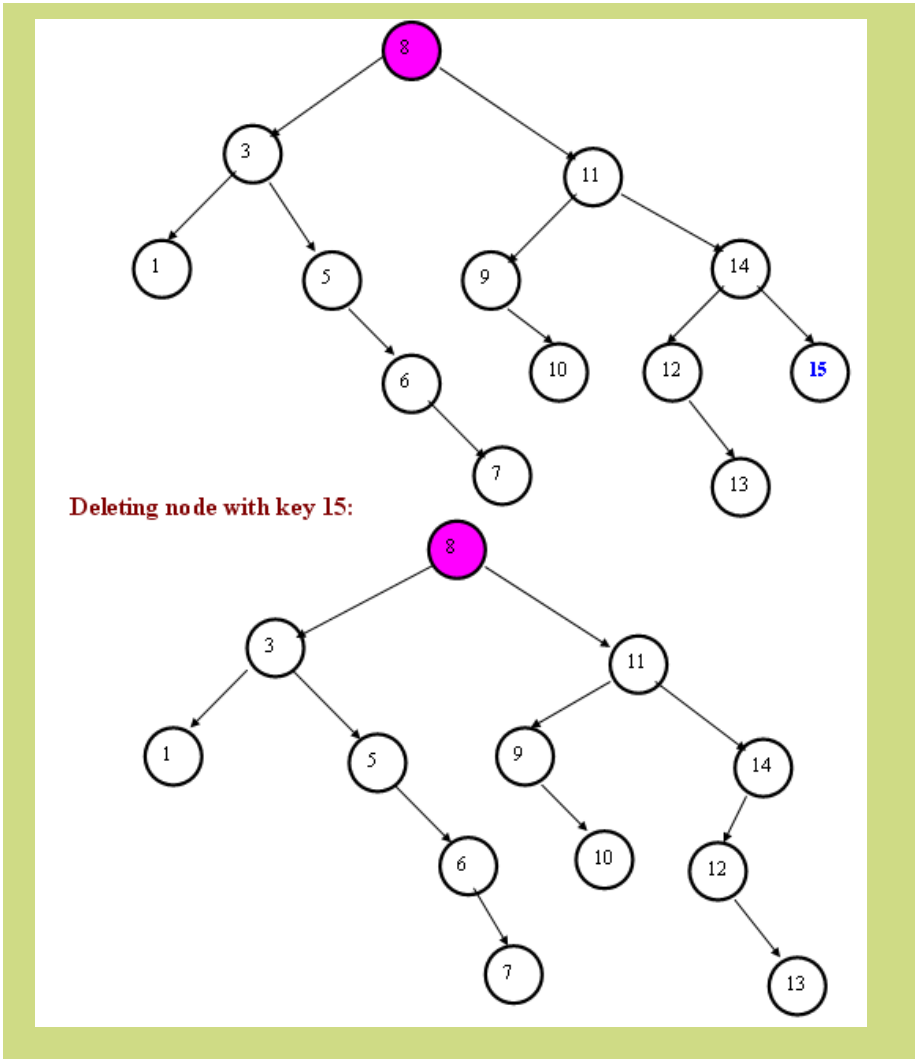
Write the prolog and fill up all information for this exercise as given in the sample. Submit the source code, input and the output files. The program is expected to be well commented. Place your program as soft copy on assigned shared drive for students of this course.

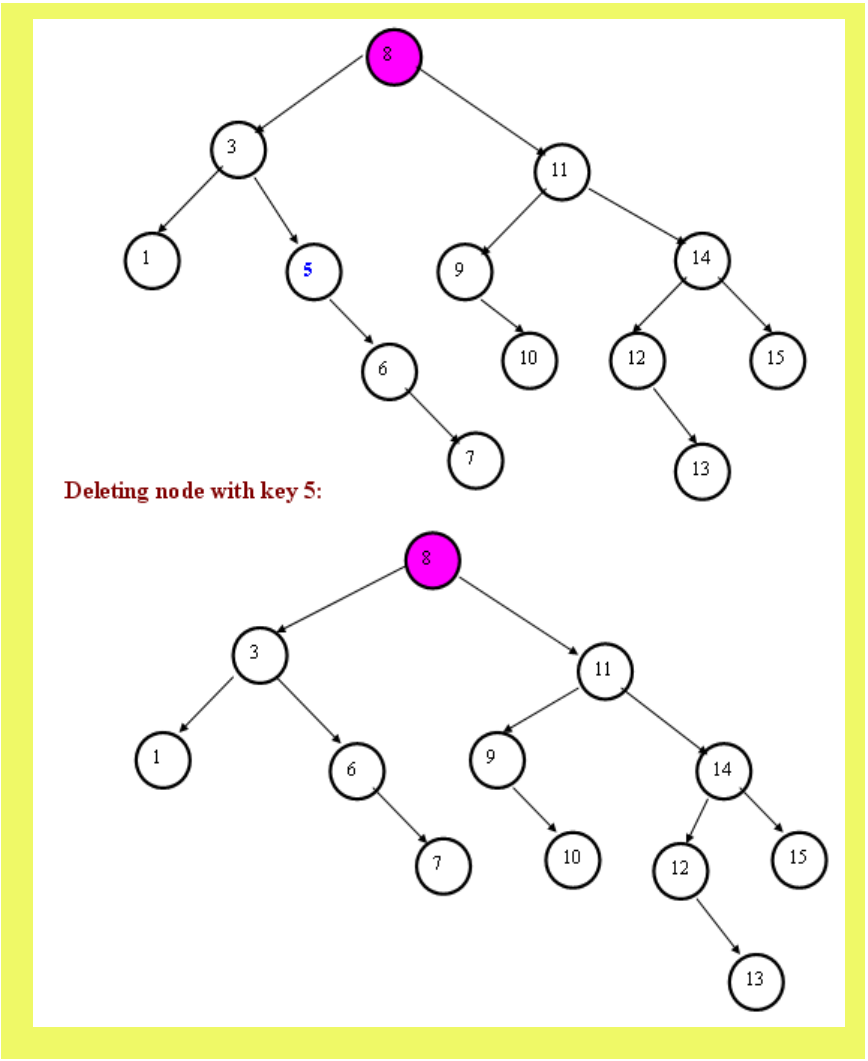
DUE DATES

Assignments are due on the following week after completing the chapter discussion.

EXERCISE 13

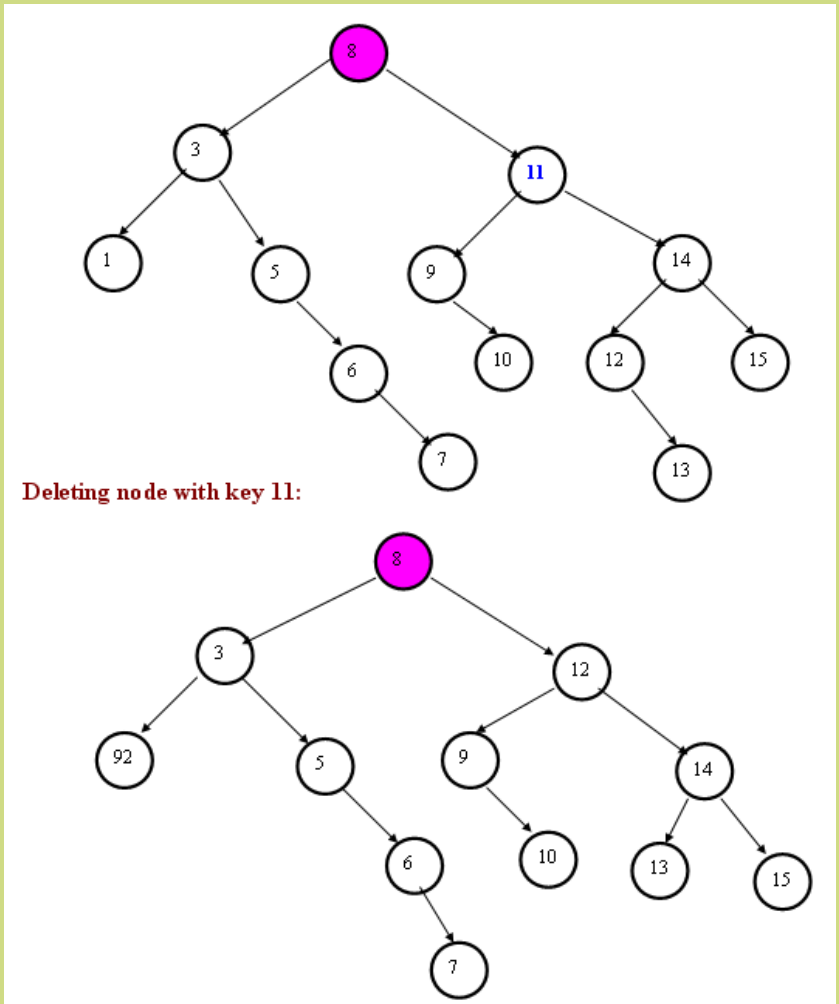
DELETING A NODE FROM BINARY SEARCH TREE





EXERCISE 13

DELETING A NODE FROM BINARY SEARCH TREE



Algorithm for deleting nodes:

```
p = tree;
q = null;
/* Search for the node with the key key, set
p to point to the node and q to its parent, if
any. */
while ( p != null && k(p) != key)
{
        q = p;
        p = ( key < k(p) )   ?  left(p)   :
                        right(p);
}       /* end of while     */
if (p == null)
 /* The key does not exists in the tree,
leave the tree unchanged */
    return;
```

EXERCISE 13

DELETING A NODE FROM BINARY SEARCH TREE

```
  /*set the variable rp to the node that will replace node (p).
first two cases: the node to be deleted has at most one
child.  */
  if (left (p) == null )
   rp = right (p);
  else
      if (right (p) == null )
          rp = left (p);
      else
   {/* third case: node(p) has two children. Set rp to the
inorder successor of p and f to the parent of rp. */
          f   =  p;
          rp =  right (p);
          s   =  left (rp); // s is left child of rp  */
          while (s !=    null)
          {
            f   =  rp;
            rp =  s;
            s   =  left (rp);
          } /* end of while  */
```

```
          if (f != p)   // at this point, rp is the
    inorder successor of p   */
          {
             left (f ) = right (rp); /*p is not
    parent of rp, set it to left(p) */
             right (rp) == right (p);
    /* remove node rp and replace      */
          }     /* end of if (f != p)  */
          left (rp) = left (p); /* set left child
    of rp, rp takes place of p   */
      }    /* end of else  */
      /* insert node(rp) into position formerly
    occupied by node(p)   */
       if (q == null)      // node(p) was the root
          tree = rp;
       else
          (p == left (q) )  ?  left(q) = rp  :
                               right (q) = rp;
      freenode (p);
return;
```