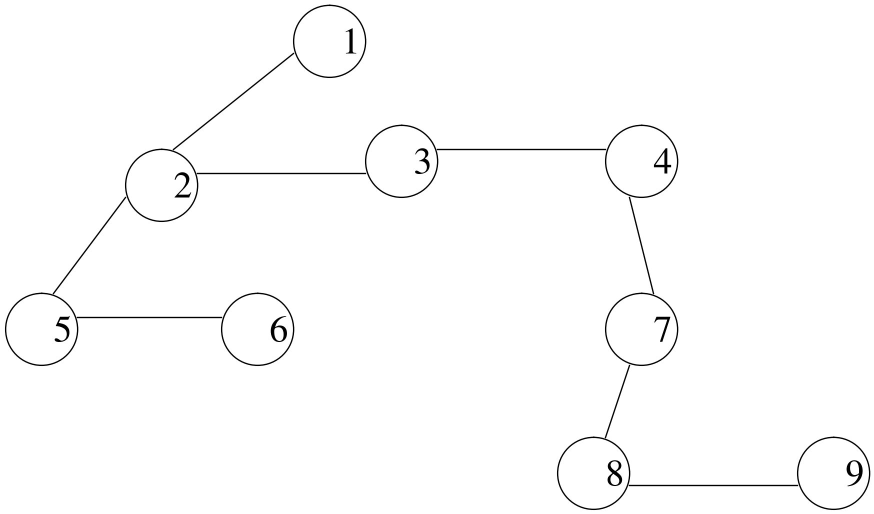



```
/*  function Height() returns the height of the tree rooted at node n.
    If the tree contains only one node, the tree's height is 1. */
int Height(node *n)
{
    if (n==NULL)
        return 0;
    else if ( (n->left_most_child==NULL) && (n->right_sibling==NULL) )
        return 1;
     else
          return Max( Height(n->left_most_child)+1,  Height(n->right_sibling) );
 }
```

*Note: function Max(a, b) returns a if a>=b, else returns b.

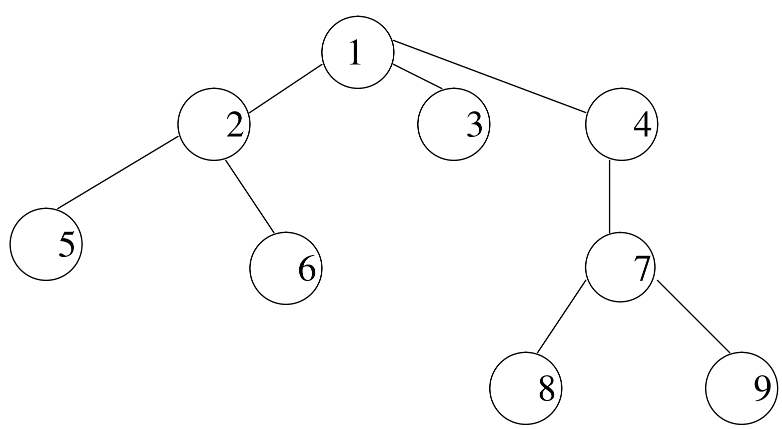
Sketch Solution of Homework 5 -- Problem 1, 2

1. Each node in the tree is represented as the following:

```
struct node {
        struct node * left;
        struct node *right;
        Datatype data;      /* Datatype indicates the type the data. */
       } Node;

/* the input of function CountLeaves() is a tree pointed by a Node pointer;
   the output of function CountLeaves() is the number of leaves in the
   tree.  */
int CountLeaves( Node *t)
{
    if (t==NULL)
         return 0;
    else if ( (t->left==NULL) && (t->right==NULL) )   /* leaf case */
         return 1;
    else
         return CountLeaves(t->left)+CountLeaves(t->right);
}
```

2. A example of left_most_child, right_sibling representation is the following:
   Let's say a tree of height 4 is like this:



Then, its corresponding left_most_child, right_sibling representation is: