AP Data Structures Name -
Ch. 6 Worksheet #9

Using the author's Ch. 6 algorithms for finding and removing nodes of a binary search tree, fill in the missing code for the find and remove functions below.

```
struct node
{
      int data;
      node * left;
      node * right;
};

node * find(node * root, const int &target)
{
      if (!root)
            return ___________________________________ ;

      if (target = = ___________________________________ )

            return root;

      else if (target < ___________________________________ )

            return ___________________________________;

      else
            return ___________________________________;
}


bool remove(node * &p, const int &data)
{
      if (!p)
      {
            return false;
      }

      if (data = = p->data)
      {
                                          // Case 1 - p has no children

            if (p->right == NULL && p->left == NULL)
            {
                  node *x = p;
                  p = 0;
                  delete x;
                  return true;
            }

                 // Case 2 - p has a right child that has no left child

            if (_____________________________________)
            {
                  node *x = p;
                  p = x->right;
                  delete x;
```

```
            return _____________________;
        }

            // Case #3 - p has a left child w/ no right child
            //           (Subcase #1 at top of p. 409)
        if (_____________________________)
        {
            node *x = p;
            p = x->left;
            p->right = x->right;
            ___________________________________

            ___________________________________
        }

            // Case #3 - p has left child that has a right child
            //                (Subcase #2 at bottom of p. 409)
        else
        {
            node *x = p;
            node *q = x->left->right;
            node *qParent = x->left;

            while (q->right != 0)
            {
                ___________________________________


                ___________________________________
            }

            q->right = x->right;
            p = q;
            qParent->right = q->left;
            q->left = x->left;
            delete x;
            return true;
        }
    }
    else if (data < p->data)      // finding node to remove
    {
        return _____________________________________
    }
    else
    {
        return _____________________________________
    }

}
```