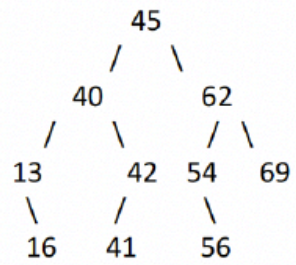
CS 300 Data Structures
Problem Set #17: Binary Search Tree

1. Draw the **Binary Search Tree** that results when the following integer values are inserted into an initially empty tree in the order shown below

   45 40 62 13 54 56 69 42 41 16

a) [10 points] Draw the Binary Search Tree



b) [5 points] Preorder traversal of the tree: 45 40 13 16 42 41 62 54 56 69

c) [5 points] Inorder traversal of the tree: 13 16 40 41 42 45 54 56 62 69

d) [5 points] Postorder traversal of the tree:16 13 41 42 40 56 54 69 62 45

2. Write a **recursive** member function to the `BinarySearchTree` header file, that calculates the `sum` of the values in the tree

```
int BST::sum(){
   return sum(root);
}
Int BST::sum(BinaryNode* p){
      if(p==NULL)
            return 0;
     else
            return p->data + sum(p->left) + sum(p->right);
}
```

CS 300 Data Structures
Problem Set #17: Binary Search Tree

3. Write a **recursive** member function to the `BinarySearchTree` header file, named `count_parents`, which finds and returns the number of parent nodes (having 1 or 2 children) in the tree.

Please make necessary modifications to `BinarySearchTree` header file so that the following sample call could be done:

```
BinarySearchTree<int> tree; //creates a binary search tree object
…                           //assume that a set of numbers inserted into the tree
cout<<tree.count_parents(); //prints out the number of parents
                            //in the binary search tree created
```

```
int BST::count_parents(){
   return count_parents(root);
}
Int BST:: count_parents(BinaryNode* p){
      if(p==NULL)
              return 0;
     else if ((p->left != NULL) || (p->right != NULL))
              return 1 + count_parents (p->left) +
                                 count_parents (p->right);
     else
              return 0;

}
```