btNode.cpp

```
void bst_insert(btNode* &bst_root, int insInt) {
   if (bst_root == 0) bst_root = create_node(insInt);
   btNode* ref = bst_root;
   btNode* pRef = bst_root;
   while (ref != 0) {
      if (ref->data > insInt) {
         if (ref->left != 0) {
            pRef = ref;
            ref = ref->left;
         }
         else {
            ref-> left = create_node(insInt);
            return;
         }
      }
      else if (ref->data < insInt) {
         if (ref->right != 0) {
            pRef = ref;
            ref = ref->right;
         }
         else {
            ref-> right = create_node(insInt);
            return;
         }
      }
      else {
         ref->data = insInt;
         return;
      }
   }

}

btNode* create_node(int data) {
   btNode* newNode = new btNode;
   newNode->data = data;
   newNode->left = newNode->right = 0;
   return newNode;
}

bool bst_remove(btNode* &bst_root, int data) {
   if (bst_root == 0) return false;
   if (bst_root->data > data) return bst_remove(bst_root->left, data);
```

```
   if (bst_root->data < data) return bst_remove(bst_root->right, data);
   if (bst_root->data == data) {
      if (bst_root->right == 0 || bst_root->left == 0) {
         btNode* old_bst_root = bst_root;
         if (bst_root->right == 0) bst_root = bst_root->left;
         else bst_root = bst_root->right;
         delete old_bst_root;
         return true;
      }
      else {
         bst_remove_max(bst_root->left, bst_root->data);
         return true;
      }
   }
   return false;
}

void bst_remove_max(btNode* &bst_root, int &data) {
   if (bst_root->right != 0) bst_remove_max(bst_root->right, data);
   else {
      data = bst_root->data;
      btNode* old_bst_root = bst_root;
      bst_root = bst_root->left;
      delete old_bst_root;
   }
}
```

btNode.h (included bc I also added the createNode() function)

```
// pre:  bst_root is root pointer of a binary search tree (may be 0 for
//       empty tree)
// post: If no node in the binary search tree has data equals insInt, a
//       node with data insInt has been created and inserted at the proper
//       location in the tree to maintain binary search tree property.
//       If a node with data equals insInt is found, the node's data field
//       has been overwritten with insInt; no new node has been created.

void bst_insert(btNode* &bst_root, int insInt);

// pre:  bst_root is root pointer of a binary search tree (may be 0 for
//       empty tree)
// post: If remInt was in the tree, then remInt has been removed, bst_root
//       now points to the root of the new (smaller) binary search tree,
//       and the function returns true. Otherwise, if remInt was not in the
//       tree, then the tree is unchanged, and the function returns false.
```

```
bool bst_remove(btNode* &bst_root, int insInt);

// pre:  bst_root is root pointer of a non-empty binary search tree
// post: The largest item in the binary search tree has been removed, and
//       bst_root now points to the root of the new (smaller) binary search
//       tree. The reference parameter, removed, has been set to a copy of
//       the removed item.

void bst_remove_max(btNode* &bst_root, int &data);

// pre:
// post: creates a node filled with the input data and right/left paths set to 0

btNode* create_node(int data);
```