```
/*
 * Class name:  BSTNode.h
 * Class description:  Templated node for the associated binary search tree class (BSTree.h).  Stores a
 * single value and contains pointers to two child nodes, where the left child is less than the contained value
 * and the right child is greater than the contained value.
 *
 * Programmer:  Chad Philip Johnson
 * Date Created:  Friday, April 26th, 2013
 * Last Date Modified:  Thursday, May 09th, 2013
 *
 * Sources Used:
 *      N/A
 */

using namespace std;

#ifndef BSTNODE_H
#define BSTNODE_H

template<typename T>
class BSTNode
{
    public:
        /**
         * Default constructor for the BSTNode class.  Sets the left child and right child pointers to NULL and
         *      assigns a given value to the node.
         * @param tmpData The value to be assigned to the node.
         */
        BSTNode( T tmpData )
        : tmpData( tmpData ), pobjBSTNodeLeftChild( NULL ), pobjBSTNodeRightChild( NULL )
        {
            /* empty */
        }

        /**
         * Destructor for the BSTNode class.  Currently unused.
         */
        virtual ~BSTNode()
        {
            /* empty */
        }

        /**
         * Changes the value associated with the current node.
         * @param tmpData The value to be assigned to the node.
         */
        void setData( T tmpData )
        {
            this->tmpData   = tmpData;
        }
```

```
        /**
         * Changes the associated left node (value is less than current node) with the current node as its
         *      left child.
         * @param pobjBSTNodeLeftChild The pointer to the left child node.
         */
        void setLeftChild( BSTNode* pobjBSTNodeLeftChild )
        {
            this->pobjBSTNodeLeftChild  = pobjBSTNodeLeftChild;
        }

        /**
         * Changes the associated right node (value is greater than current node) with the current node as its
         *      right child.
         * @param pobjBSTNodeRightChild The pointer to the right child node.
         */
        void setRightChild( BSTNode* pobjBSTNodeRightChild )
        {
            this->pobjBSTNodeRightChild = pobjBSTNodeRightChild;
        }

        /**
         * Return the value of the node.
         * @return The value of the node.
         */
        T getData() const
        {
            return this->tmpData;
        }

        /**
         * Return the memory address of the variable storing the value of the node.
         * @return The memory address of the variable storing the value of the node.
         */
        T& getData()
        {
            return this->tmpData;
        }

        /**
         * Return the pointer value for the associated left child node.
         * @return The pointer value for the left child node.
         */
        BSTNode<T>* getLeftChild() const
        {
            return this->pobjBSTNodeLeftChild;
        }

        /**
         * Return the memory address of the pointer for the associated left child node.
```

```
         * @return The memory address of the pointer for the associated left child node.
         */
        BSTNode<T>*& getLeftChild()
        {
            return this->pobjBSTNodeLeftChild;
        }

        /**
         * Return the pointer value for the associated right child node.
         * @return The pointer value for the right child node.
         */
        BSTNode<T>* getRightChild() const
        {
            return this->pobjBSTNodeRightChild;
        }

        /**
         * Return the memory address of the pointer for the associated right child node.
         * @return The memory address of the pointer for the associated right child node.
         */
        BSTNode<T>*& getRightChild()
        {
            return this->pobjBSTNodeRightChild;
        }

    private:
        BSTNode<T>* pobjBSTNodeLeftChild;
        BSTNode<T>* pobjBSTNodeRightChild;
        T tmpData;

};

#endif
```