Initial "UML" for **bst** template class

adapted from Ch. 10, Savitch and Main, "Data Structures and Other Objects Using C++"

Template Class: `bst<Item>`

/* a binary search tree where each node contains an Item.
For this version, it is assumed that all nodes contain different-valued Items. */

Member data and related details:

* contains elements of type `value_type`; this is set to be the value of template parameter **Item**
* has a size of `size_t`

Constructors:

/* postcondition: creates an empty **bst** instance (with no nodes) */

```
bst( );
```

Accessors and other observer member functions:

/* postcondition: returns the number of nodes in the bst. */

```
size_t       get_size( )         const;
```

/* postcondition: returns **true** if bst is empty, and returns **false** otherwise */

```
bool         is_empty( ) const;
```

/* postconditions: returns true if **entry** is in this bst, and returns false otherwise */

```
bool         contains(const Item& entry);
```

Modifiers and other modifying member functions:

/* postconditions: IF **entry** is already in the bst, the bst is unchanged.
otherwise, **entry** is added to the bst in such a way that the binary search
tree properties still hold. */

```
void         add(const Item& entry);
```

/* postconditions: prints the current contents of the bst, 1 per line, as they result from an in-order traversal. */

```
void         print_inorder( );
```