```
#include "treasurechest.h"

void TreasureChest::addItem (const Item& newItem)
{
    this->chest.push_back( newItem );
}

/*
 * Insert an item at the specified zero-indexed position in the chest.
 * If position is not valid for the chest, add the item to
 * the end of the chest.
 * @param newItem the item to be inserted into the chest
 * @param position the zero-indexed position where the insertion
 *        is to take place
 */
void TreasureChest::insertItem (const Item& newItem, unsigned int position)
{
    if( position < chest.size() )
    {
        vector<Item>::iterator it = this->chest.begin();;
        this->chest.insert( (it + position), newItem );
    }
    else
    {
        this->addItem( newItem );
    }
}

/*
 * Get a pointer to an item at a specified zero-indexed position in the chest.
 * @param position the zero-indexed position of the item
 * @return a pointer to the item if position is valid, else NULL
 */
const Item* TreasureChest::getItem (unsigned int position)
{
    if( position < chest.size() )
    {
        return &this->chest.at( position );
    }
    else
    {
        return NULL;
    }
}

/*
 * Remove an item from the chest at a specified zero-indexed position.
 * @param position the zero-indexed position of the item
 * @return a copy of the Item removed from the chest
 * @throw string("ERROR: attempting remove at invalid position") if
```

```
 *        position is not valid
 */
Item TreasureChest::removeItem (unsigned int position) throw (string)
{
    if( position < chest.size() )
    {
        vector<Item>::iterator it   = this->chest.begin();;
        Item objRemovedItem         = chest.at( position );
        this->chest.erase( (it + position) );

        return objRemovedItem;
    }
    else
    {
        throw string( "ERROR: attempting remove at invalid position" );
    }
}

/*
 * Clear the chest of all items.
 */
void TreasureChest::clear ()
{
    this->chest.clear();
}

/*
 * Check to see if the chest is empty.
 * @return true if the chest is empty, else false
 */
bool TreasureChest::empty () const
{
    if( this->chest.empty() )
    {
        return true;
    }

    return false;
}

/*
 * Get the size/number of items currently in the chest.
 * @return an unsigned integer containing the current size of the chest
 */
unsigned int TreasureChest::getSize () const
{
    return this->chest.size();
}

/*
```

```
 * Sort the items in the chest by name, using the sort function
 * from the C++ standard algorithm library.
 */
void TreasureChest::sortByName ()
{
    sort( this->chest.begin(), this->chest.end(), compareItemsByName );
}

/*
 * Sort the items in the chest by value, using the sort function
 * from the C++ standard algorithm library.
 */
void TreasureChest::sortByValue ()
{
    sort( this->chest.begin(), this->chest.end(), compareItemsByValue );
}

/*
 * Sort the items in the chest by quantity, using the sort function
 * from the C++ standard algorithm library.
 */
void TreasureChest::sortByQuantity ()
{
    sort( this->chest.begin(), this->chest.end(), compareItemsByQuantity );
}

/*
 * Place the names of the items in the chest on the specified stream,
 * formatted as ITEM_NAME,ITEM_NAME,...ITEM_NAME
 */
ostream& operator<< (ostream& outs, const TreasureChest& src)
{
    for( vector<Item>::const_iterator p = src.chest.begin(); p < src.chest.end(); p++ )
    {
        outs << *p;
        if( (p + 1) != src.chest.end() )
        {
            outs << ",";
        }
    }

    return outs;
}

/*
 * Compare two items by name.
 * @return true if lsrc.name < rsrc.name, else false
 */
bool compareItemsByName (const Item& lsrc, const Item& rsrc)
{
```

```
    return ( lsrc.name < rsrc.name );
}

/*
 * Compare two items by value.
 * @return true if lsrc.value < rsrc.value, else false
 */
bool compareItemsByValue (const Item& lsrc, const Item& rsrc)
{
    return ( lsrc.value < rsrc.value );
}

/*
 * Compare two items by quantity.
 * @return true if lsrc.quantity < rsrc.quantity, else false
 */
bool compareItemsByQuantity (const Item& lsrc, const Item& rsrc)
{
    return ( lsrc.quantity < rsrc.quantity );
}

```