

2001 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. Consider the following declarations for maintaining a list of books. Information about each book includes the title, author, and an appropriate age range for readers. The list is ordered by age range, as defined by function `LessThan`. Assume that a book appears at most once in the list.

```
struct Book
{
    apstring title; // title of book
    apstring author; // author of book
    int lowAge;      // lowest recommended age
    int highAge;     // highest recommended age
};

bool LessThan(const Book & lhs, const Book & rhs);
// postcondition: returns true if lowAge of lhs < lowAge of rhs or
//                if lowAge of lhs and rhs are equal
//                and highAge of lhs < highAge of rhs;
//                otherwise, returns false

class BookList
{
public:
    BookList(); // constructor

    void InsertOne(const Book & bk);
    // precondition: this BookList is in sorted order by age range
    //                as defined by LessThan;
    //                bk is not already in this BookList
    // postcondition: bk has been inserted into this BookList,
    //                maintaining its order by age range

    void InsertMany(const apvector<Book> & second);
    // precondition: this BookList is in sorted order by age range
    //                as defined by LessThan; second contains
    //                second.length() books in arbitrary order;
    //                none of the books in second are in this BookList
    // postcondition: all the books from second have been inserted into
    //                this BookList, maintaining its order by age range

    // ... other public member functions not shown

private:
    apvector<Book> myList;
    // collection of books in sorted order as defined by LessThan;
    // myList.length() > 0

    int myCount;
    // number of books in myList
};
```

2001 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the free function `LessThan`, as started below. `LessThan` returns `true` if either
- `lowAge` of the first book is less than `lowAge` of the second book; or
 - `lowAge` is the same for both books, and `highAge` of the first book is less than `highAge` of second book.

Otherwise, `LessThan` returns `false`.

For example:

BookA		BookB		LessThan(BookA, BookB)
lowAge	highAge	lowAge	highAge	
9	12	9	14	true
9	12	10	11	true
9	12	10	15	true
9	12	8	15	false
9	12	9	11	false
9	12	9	12	false

Complete function `LessThan` below.

```
bool LessThan(const Book & lhs, const Book & rhs)
// postcondition: returns true if lowAge of lhs < lowAge of rhs or
//                if lowAge of lhs and rhs are equal
//                and highAge of lhs < highAge of rhs;
//                otherwise, returns false
```

2001 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `BookList` member function `InsertOne`, as started below. Assume that the private data member `myList` is already ordered as defined by `LessThan`. `InsertOne` places a book into the `BookList`, maintaining that order, and will resize `myList` by doubling the capacity, if necessary.

For example, assume that `BookList library` contains the following books.

title	author	lowAge	highAge
Madeline	Bemelmans	3	8
The Lorax	Seuss	3	10
Harry Potter and the Sorcerer's Stone	Rowling	9	99
Holes	Sacher	12	18
I Know Why the Caged Bird Sings	Angelou	16	99

Consider the following `Book bk` to be inserted into `library`.

title	author	lowAge	highAge
Little House on the Prairie	Wilder	8	12

After the call `library.InsertOne(bk)`, `library` contains the following books. Note that the books are in sorted order by `lowAge`, then by `highAge` within `lowAge`.

title	author	lowAge	highAge
Madeline	Bemelmans	3	8
The Lorax	Seuss	3	10
Little House on the Prairie	Wilder	8	12
Harry Potter and the Sorcerer's Stone	Rowling	9	99
Holes	Sacher	12	18
I Know Why the Caged Bird Sings	Angelou	16	99

In writing `InsertOne`, you may call function `LessThan` specified in part (a). Assume that `LessThan` works as specified, regardless of what you wrote in part (a).

Complete function `InsertOne` below.

```
void BookList::InsertOne(const Book & bk)
// precondition:  this BookList is in sorted order by age range
//               as defined by LessThan;
//               bk is not already in this BookList
// postcondition: bk has been inserted into this BookList,
//               maintaining its order by age range
```

2001 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) Write the `BookList` member function `InsertMany`, as started below. `InsertMany` will insert all the books from an array of books into this `BookList`, maintaining the sorted order of the private data member `myList`, as defined by `LessThan`.

For example, assume that the array `books` contains the following list of books to be inserted into the initial version of `library` shown in part (b).

	title	author	lowAge	highAge
0	The Phantom Tollbooth	Juster	9	12
1	Invisible Man	Ellison	15	99
2	Charlotte's Web	White	8	12

The following table shows the contents of `library` after the call `library.InsertMany(books)`.

	title	author	lowAge	highAge
	Madeline	Bemelmans	3	8
	The Lorax	Seuss	3	10
	Charlotte's Web	White	8	12
	The Phantom Tollbooth	Juster	9	12
	Harry Potter and the Sorcerer's Stone	Rowling	9	99
	Holes	Sacher	12	18
	Invisible Man	Ellison	15	99
	I Know Why the Caged Bird Sings	Angelou	16	99

In writing `InsertMany`, you may call functions `LessThan` and `InsertOne` specified in parts (a) and (b). Assume that `LessThan` and `InsertOne` work as specified, regardless of what you wrote in parts (a) and (b). Do not assume that the list of books to be added is in any particular order.

Complete function `InsertMany` below.

```
void BookList::InsertMany(const apvector<Book> & second)
// precondition:  this BookList is in sorted order by age range
//               as defined by LessThan; second contains
//               second.length() books in arbitrary order;
//               none of the books in second are in this BookList
// postcondition: all the books from second have been inserted into
//               this BookList, maintaining its order by age range
```