COP 3502H Assignment #5
Binary Tree Book Store
Assigned: Tuesday 4/10/2012
Due: Friday 4/20/2012, 11:55pm on Webcourses

The Problem: The Book Store

You have just opened a new book store and want to monitor your inventory. You will store your inventory of books in a Binary Search Tree by inserting your current inventory into the tree at the beginning of the day. As books are sold throughout the day you will update their quantities, and if a book sells out you will delete that book from the tree.

Implementation Specifications

Since the Binary Search Tree will store Books, you will use the following struct to store the Book information:

```
typedef struct  {
    int key;
    char title[50];
    char author[30];
    int quantity;
} Book;

typedef struct  {
    node *left;
    node *right;
    Book *bk;
} node;
```

- Note when implementing delete, when deleting a node that has 2 children replace with the max in the left subtree.
    - When deleting a node that has exactly one child node, then replace the node to delete with the sole child node.
- The tree nodes should be stored using the Binary Search Tree property, where the book's key is compared to determine whether the node belongs in the left or the right subtree of the root.

Your program will read in input from the file ***"books.in"*** and then output each corresponding solution to the screen.

Input File Format

The first line of the input file will contain a single positive integer *n*, representing the number of test cases in the file. Each test case will follow, one by one. For each test case, the first line will contain a single positive integer *m*, representing the number of operations that will occur. There will be a series of *m* operations of the following types: POPULATE and SOLD. The POPULATE operation will be followed by the key, quantity, author, and title. The SOLD operation will be followed by the key and quantity. All author names and titles will only contain alphanumeric characters and underscores, no spaces. Your tree should be sorted by the key, thus you should compare keys when inserting a new book.

Deliverables

1) The tree implementation, **`BinarySearchTree.h`** (optional, BinarySearchTree.c, if you want your function implementations in a separate file)
2) **`BookStore.c`** which reads input from the file and creates the tree.

Output File Format (Write to "books.out")

For each test case, output the contents of the Binary Search Tree using ***in order traversal*** to print the element key, title and quantity. Finally, print which items sold out in the order they sold out during the day. For example:

```
Test Case #X:
     Element #1. BookTitle1  quantity1
     Element #2. BookTitle8  quantity8
The book BookTitle_blah sold out.
The book BookTitle_etc sold out.
```

Sample Input File

```
2
10
POPULATE 5 10 JD_Salinger Catcher_in_the_Rye
POPULATE 2 20 Isabel_Allende La_Casa_De_Los_Espiritus
POPULATE 3 30 Ken_Follet Pillars_of_the_Earth
POPULATE 6 8 Herman_Wouk War_and_Remembrance
POPULATE 7 3 Margaret_Mitchell Gone_with_the_Wind
POPULATE 1 5 Emily_Bronte Wuthering_Heights
SOLD 5 5
SOLD 2 10
SOLD 7 3
SOLD 5 5
7
POPULATE 1 10 AB C
POPULATE 3 10 DE F
POPULATE 2 10 GH I
POPULATE 5 10 JK L
SOLD 5 5
SOLD 5 5
```

```
SOLD 2 7
```

Sample Output

```
Test Case #1:
      Element #1. Wuthering_Heights 5
      Element #2. La_Casa_De_Los_Espiritus 10
      Element #3. Pillars_of_the_Earth 30
      Element #6. War_and_Remembrance 8
The book Gone_with_the_Wind by Margaret_Mitchell sold out.
The book Catcher_in_the_Rye by JD_Salinger sold out.

Test Case #2:
      Element #1. C 10
      Element #2. I 7
      Element #3. F 10
The book L by JK sold out.
```