Program 4 {August 18, 2014} 45 Points
Robot Deliveries Database
Due: Friday, October 3, 2014 at 11:59 p.m.

This program focuses on using tree data structures to store information for subsequent retrieval. This assignment is to be done in C++ by two-person assigned teams and the use of binary trees or another approved data structure is **required** for this assignment.

Working from Program 2, assume robots visit a series of stores delivering items to the stores. Once a robot has completed a delivery to a store, the robot sends a record of the item identifiers and respective quantities delivered to that store to an output log file. Each subsequent robot in turn appends its delivery information to the same log file. This program uses the robot log file as input.

The objective of this program is to read in a single stream of recorded robot deliveries (see assignment input) and build a shoppers' database that is a **binary tree** of all objects currently in the RoboMall. Initially, the RoboMall contains no items in any of its stores. All items are identified by a two-digit ASCII code where the first digit is a **capital** letter of the alphabet and the second digit is a number (between 0 and 9). The binary tree keeps a record of the items delivered by the robots in alphabetical order by code and records in a list (probably a linked list) the stores that currently have that item to sell and the cumulative count of the number of available items at that store. The per-item list is kept in **decreasing count order** based on the number of items a store currently has. For example, if the robots deliver 6 L7's to store (4, 4, 0) and subsequently delivers 14 L7's to store (12, 6, 1), then an entry in the L7 list for store (12, 6, 1) would come before the entry in the L7 list for store (4, 4, 0). Once your program has processed all the robot delivery information, it outputs **ALL** the information in the database in alphabetical order by item including the sorted list for each item.

Assignment Input

The program reads in robot delivery information from an ASCII file. To keep it simple, do **NOT** use C file I/O commands. The file should be accessed by using a Linux command line that **redirects** standard input from a file. This enables testing the program either using a file or terminal input.

The input file consists of a series of lines of input in the following format:

```
x-pos y-pos z-pos  k-items  item1 count1  item2 count2 … itemk countk
```

where

x-pos y-pos z-pos	is the coordinates for the store receiving a robot delivery.
k-items	is the number of unique items reported on this line of input.
$item_i$	is the two digit ASCII code for the i^{th} item on the delivery list.
$count_i$	is the quantity of $item_i$ delivered to the store.

For example, the following input line:

```
4 4 0 3 L7 6 C8 100 T4 19
```

indicates that a robot delivered to a store located at row 4, column 4 on the First Floor three sets of items: 6 items coded as L7, 100 items coded as C8 and 19 items coded as T4.

The program continues reading in data from the file until it encounters the line:

```
16 16 16
```

This illegal entry serves as a **sentinel** to indicate the end of the input for the robot delivery file.

Once the input file has been completely read and all items properly added to the binary tree database, the program prints out the **complete** database of items in alphabetical order with all the data recorded sent to the redirected output file `prog4.out`.

What to turn in for Program 4 (prog4)

An official test input file will be made available a few days before the due date. Turn in your assignment using the **turnin** program on the CCC machines. You should turn in a tarred file that includes your source code, a make file, a README file and prog4.out. While teammates may work together on one specific subroutine or function, it is required for Program 4 that the comments in the source code list ONLY A SINGLE AUTHOR per piece of code. If two people worked equally on several routines then divide up the authorship in an equitable manner that reflects the effort fairly while still assigning only one author per routine. This is necessary to provide a rough estimate of the amount of effort each team member contributed to Program 4. The README file should provide any information to help the TA or SA test your program for grading purposes. The README file must also identify parts of your program that are not working correctly when you turned it in.