**Lab 15**

**Binary Files**

This lab is somewhat similar to lab 14. We need a program to help manage an inventory. Each part in the inventory has a description, a current quantity on hand, and a maximum quantity of that part to stock (both quantities are within the range 0 to 1,000 inclusive). The inventory is stored on disk in a binary file. Upon startup, the file is opened for reading and writing, but nothing is immediately read.

When the program is running, the user is prompted to choose a part number. That part (and *only* that part) is then read by random access IO from the disk file and displayed to the user. If the current quantity is less than the maximum quantity, the user can choose whether or not to restock the part. If the user chooses to restock, enough parts are added to the inventory to bring the current quantity up to the maximum quantity, and the new part information is immediately written to the file, again using random access IO. If the user does not restock, the disk file is not changed.

I am providing you with three programs. You already have dump.cpp which shows the contents of any file. [Here](http://borax.truman.edu/180/lab15bin/create_binary_inventory.cpp) is a program that creates a binary inventory file on disk. Running this program assures that you have a valid, uncorrupted file to start with. Do not change the program in any way. Run this program when you need to make a clean, uncorrupted binary data file. When you run the program, it creates this inventory:

Thermistor 703 950

Thyratron 117 185

Inclinometer 670 770

Rectifier 312 545

Darlington NPN 11 15

Yagi antenna 205 930

MOSFET 43 100

Nixie tube 333 900

DIAC 115 760

Klystron 3 10

Stepping motor 117 510

Piezoelectric crystal 154 820

The [framework.cpp](http://borax.truman.edu/180/lab15bin/framework.cpp) program prompts the user for a part number and displays that part. Your assignment is to enhance this program so that it also prompts, if appropriate, for the user to choose to restock, and restocks the part if the user so requests. Just to be clear, *do not* create a vector of parts in this program. The disk file is the database, not a vector in RAM. You will have to write a function named to\_string that accepts a Part as its parameter for the framework to compile (I have provided the Javadoc for this function).

Create the function to\_string so that the framework will compile, and run it to see how it works (after running create\_binary\_inventory). Then, extend the program to complete the lab assignment.

A run of your finished program should look like this.

$ ./program

Choose part number or q to quit (0 to 11): 4

4 Darlington NPN 11 15

Restock? (Y/N) n

Choose part number or q to quit (0 to 11): 6

6 MOSFET 43 100

Restock? (Y/N) y

Restocking MOSFET adding 57

6 MOSFET 100 100

Choose part number or q to quit (0 to 11): 6

6 MOSFET 100 100

Already at maximum quantity

Choose part number or q to quit (0 to 11): q

If you now run the program again, Darlington NPNs should still have a current quantity of 11, and the MOSFETs should have a current quantity of 100.

The simplest way for your program to update the disk file is to overwrite the entire part if the current part has been changed. For extra credit, only overwrite the current quantity of the current part, because the description and max quantity never change.

As always, carefully read the style guide and make sure your program conforms to it.

When you are satisfied with your program, by the due date of 5 pm Saturday, 7 December, submit it in the usual way.