CMPS 12B

Introduction to Data Structures

Programming Assignment 1

Objective

The purpose of this assignment is to give you practice in working with sorted arrays (binary search) and with implementing sorting algorithms (merge sort).

Task

Your task is to write a business search server. The finished product will be a Java application that is started from the command line with one argument, the filename of a business database. The database file is a text file that contains one business per line. Each line contains the business name, a comma, then a phone number

```
Bob's Pizza,(831)555-1212
```

Once the server is started it reads the entire business database file and builds an array of businesses sorted by name. It then waits for input on stdin. After a business name is entered, the server does a binary search to find the given business. If the business name is not found, the server prints "NOT FOUND". If the business is found then the server prints out its telephone number. A blank line as input causes the server to terminate.

Input File Format

You may assume your program will be tested only on files with the following formatting. The first line of the file contains exactly one integer N and nothing else. The next N lines contain one business per line. Each business line consists of a business name (any sequence of characters other than a comma), then a comma, then a phone number (any sequence of characters other than a newline), then a newline. Each business line contains exactly one comma; everything before the comma is a business name and everything after the comma is the telephone number.

Here is an example business database named favorites.txt:

```
4
Bob's Pizza,(831)555-1212
Pizza My Animal,(831)555-3231
Sous Vide Surprise,(831)555-8080
Aaron and Sons,(831)555-1000
```

Program Behavior

If run without any arguments, your program must display a help message showing the correct way to invoke the program and then terminate. If run with one argument, the single argument represents a filename of businesses. The program must open the file, read the first number, allocate an array of the appropriate size, then read in all the businesses in the database.

Your program will search for businesses using binary search, which requires the array to be in sorted order. Once all the data is read into an unordered array, your program must sort the businesses by name using mergesort so that it can find businesses using binary search.

Once the sort is finished, the server should read data from stdin. Each line that is entered represents a business name to search for. A blank line must print out how many queries were answered and how many resulted in not found, then terminate the server. A non-blank line represents a business name. Your program must use binary search to attempt to find the business in the sorted array. If found, the telephone number should be displayed. If not found, "NOT FOUND" should be displayed.

Example run:

```
$ BusinessSearch
Usage: BusinessSearch businessDB
$ BusinessSearch favorites.txt
Bob's
NOT FOUND
Bob's Pizza
(831)555-1212
Sous Vide Surprise
(831)555-8080
Sous Vide Surprise
(831)555-8080

4 total queries, 1 not found
$
```

Partial Credit

For partial credit you may simply report FOUND or NOT FOUND for each query. This simplifies the program because your array can hold a simple String containing the business name. The full version of the program must keep an array of records, where each record contains a name and a phone number. You may want to start with the simpler version and submit that, then work on also keeping track of phone numbers.

For partial credit you may use linear search on an unsorted array of businesses. This simplifies the program because you only have to read in the businesses, you don't have to sort them using merge sort. The search functionality is also simplified because it just has to scan the array until it finds the match. The external behavior of this version is exactly the same as the full version with sorting and binary search. You may want to start with this simpler version and submit it, then work on sorting your array and searching using binary search.

For partial credit you may use insertion sort instead of merge sort. No credit for bubble sort.

Command-line Arguments and File Input

If you don't know how to get command-line arguments into your program or how to read the contents of the file or stdin, don't worry. This is included in Lab 2 which is due before this assignment. You can start by manually initializing an example business database array into your program and hard-coding a specific query into your code. For example, you might start by not keeping track of phone numbers and just doing a linear search of an unsorted array and hard-coding information in the following way:

```
String[] businesses = {
      "Bob's Pizza", "Pizza My Animal", "Sous Vide Surprise", "Aaron and Sons"
};
String query = "Sous Vide Surprise";
```

Once you have completed Lab 2 you can update your code to read queries from stdin and read the businesses from the file named in the first argument to your program.

What to Submit

Submit the files README, Makefile, and BusinessSearch.java. Your makefile must make an executable jar file called BusinessSearch, and must include a target to be able to do "make clean" which removes all generated files. Remember to include a comment block on all files with appropriate information. See lab 1 for more details on this. Submit all files to the assignment name "asg1".