# Assignment 3 - 100 points

## Assignment Overview

In this assignment, you will add some more functionality to the ProviderDB class.

## Purpose

This assignment introduces searching using the binary search algorithm.

## Input

This assignment will continue to use the binary providerdb file from Assignment 2 as input. In addition, a text file containing transaction records will be used. You can make a link to this file in your assignment directory using the following UNIX command:

ln -s ~t90kjm1/CS241/Data/Fall2017/Assign3/transactions.txt

The records in this file contain data in character format. Do not try to read them as binary data.

### Makefile

The makefile for this assignment is essentially identical to the one used for Assignment 2. Only the assignment number has changed.

### Provider class

No changes to this class are required.

### ProviderDB class

Add the following two methods to the ProviderDB class

* processTransactions()

This method should take one parameter: a pointer to a constant character (that points to an array of characters contaaining the name of a file of transaction data). The method should return nothing.

This method will read a series of transaction records until end of file. Each record contains a transaction type, possibly followed by some additional data.

First, this method should read the transaction type as a string. You can easily do this using the stream extraction operator (>>). The method should then take action (or preferrably, call another method to take action) based on the transaction type:

* + Transaction Type "all"

This transaction type is not followed by any additional fields. In response to this transaction type, your program should do the following:

* + 1. Call the sortByName() method to sort the providers in ascending order by name.
    2. Call the print() method to print the sorted array of providers.
  + Transaction Type "specialty"

This transaction type is followed by a specialty. Since the specialty may contain whitespace, reading it will be a little tricky. If you try to immediately read the specialty using getline() (like in Assignment 1), you'll end up reading in the whitespace that separates the transaction type and the specialty as part of the specialty string. To deal with this, your program should do the following:

* + 1. Extract any whitespace that precedes the specialty using the ws manipulator with the stream extraction operator, e.g.:
    2. tranFile >> ws; // tranFile is the input file stream for the transaction file
    3. Read the specialty from the transaction record using getline().
    4. Call the sortBySpecialty() method to sort the providers in descending order by specialty.
    5. Loop through the sorted array of Provider objects and print any provider with a specialty that matches the specialty from the transaction record.
    6. Print a count of the number of providers with that specialty that were found
  + Transaction Type "number"

This transaction type is followed by a provider number. In response to this transaction type, your program should do the following:

* + 1. Read the provider number from the transaction record. You can read this as a string using the stream extraction operator.
    2. Call the sortByProviderNumber() method to sort the providers in ascending order by provider number.
    3. Call the searchForProviderNumber() method to search the sorted array of providers for the provider number from the transaction record.
    4. If the search returns -1, print an error message stating that the provider number was not found. Otherwise, print the Provider object with the subscript returned by the search method.
* searchForProviderNumber()

This method should take one parameter: a character array containing the provider number of the Provider to search for (searchNumber. The method should return an integer.

The logic for this method is a variation of the binary search of a sorted list strategy.

int low = 0;

int high = number of valid Provider objects in the array - 1;

int mid;

while (low <= high)

{

mid = (low + high) / 2;

if (searchNumber is equal to providerNumber data member of providerArray[mid])

return mid;

if (searchNumber is less than providerNumber data member of providerArray[mid])

high = mid - 1;

else

low = mid + 1;

}

return -1;

As usual, you'll need to use strcmp() to perform the comparison of provider numbers.

You are encouraged to write additional methods as you see fit. For example, it would make a lot of sense to split the logic of the processTransactions() method into several other methods - perhaps one to handle each type of transaction.

### Main program

The main() routine logic will for this assignment will now be extremely short and simple.

* Create a ProviderDB object using the alternate constructor you wrote. Pass the filename string "providerdb" as an argument to the constructor.
* Call the processTransactions() method for the ProviderDB object. Pass the filename string "transactions.txt" as an argument to the method.

## Other Points

* If you want to see what the records of the transaction file look like, you can open it (in read-only mode) with the nano editor.
* Remember that when reading records until end-of-file, you need to structure your loop so that you test for EOF after reading the first field on each record, and then only if that read is successful should you go on and read the remaining fields of the record inside the loop body.
* As always, programs that do not compile on turing/hopper automatically receive 0 points.
* Make sure to document your program according to the standards listed in the Course Notes book. In particular, each function should have a documentation box describing the purpose of the function, the input parameters, and the return value (if any). There should also be a documentation box for the program as a whole.
* Submit the final version of your program using the electronic submission guidelines posted on the course web site and described in class.