Department of Electrical and Computer Engineering
ECSE 202 – Introduction to Software Development
Assignment 6
A Simple Database Program

Due December 4th at 5:00 pm

Problem Description

You are given two text files, NamesIDs.txt and marks.txt containing student names and IDs in the first file and the corresponding grades in the second. The purpose of this assignment is to build a very simple database program using B-Trees for record storage. The program is run from the command line as follows:

```
> sdb NamesIDs.txt marks.txt
Building database…
Finished…

sdb:
```

The first thing the program does is to read in the student names, IDs, and grades from their respective files and store the result in a B-Tree (note, you cannot use arrays in this assignment). In the object-oriented programming lecture we discussed how to use structures and the malloc() function to create objects. Assume that our student record node is defined by the following structure:

```
struct StudentRecord {
        char first [MAXLEN];
        char last  [MAXLEN];
        int ID;
        int marks;
                struct StudentRecord *left;
                struct StudentRecord *right;
};
```

The process of building our database is to read through each file sequentially, extract the first name, last name, ID number and grade and store the result in a StudentRecord structure, which subsequently gets added to the B-Tree using the addNode() function. In fact, to keep this program as simple as possible, we'll create two B-Trees, one ordered by last name and the second ordered by student ID.

The code should look something like this:

```
struct StudentRecord  data, *rootName, *rootID;
FILE *NamesIDs;
FILE  *Marks;
```

```
if  ((NamesIDs = fopen(argv[1],”r”)) == NULL) {               // open Names IDs file
          printf(“Can’t open %s\n”,argv[1]);
          return -1;
}

if  (marks = fopen(argv[2],”r”)) == NULL) {                   // open marks file
          printf(“Can’t open %s\n”,argv[2]);
          return -2;
}

rootName=NULL;                                                // initialize 2 B-Trees
rootMarks=NULL;
int numrecords=0;

while (fscanf(NamesIDs,”%s%s%d”,                              // scan record into structure
          &(data.first[0]),
          &(data.last[0]),
          &(data.ID)) != EOF) {

          fscanf(Marks,”%d”,&(data.marks));                   // marks too
          numrecords++;

          rootName = addNode(rootName,&data);                 // copy to B-Tree sorted by last
          if (rootName==NULL) {
                    printf(“Error creating name B-Tree, aborted.\n”);
                    return -3;
          }

          rootID = addNode(rootID,&data);           // copy to B-Tree sorted by last
          if (rootID==NULL) {
                    printf(“Error creating ID B-Tree, aborted.\n”);
                    return -4;
          }

}

fclose(NamesIDs);
fclose(Marks);
```

Once the database has been built, the program runs a simple command interpreter using the following 2-letter commands:

LN	List all the records in the database ordered by last name.
LI	List all the records in the database ordered by student ID.
FN	Prompts for a name and lists the record of the student with the corresponding name.
FI	Prompts for a name and lists the record of the student with the Corresponding ID.
HELP	Prints this list
?	Prints this list
Q	Exits the program.

Note: Command input must be case insensitive (including name to be matched).

In addition to writing the addNode function, you will also have to write traversals that implement searching for name and searching for ID. The code, by the way, is almost identical to what you did in Java with the exception of how objects are allocated.

Examples:

```
> sdb NamesIDs.txt marks.txt
Building database…
Finished…

sdb: LN

Student Record Database sorted by Last Name

Dorethea Benes          2583        97
Teisha Britto           2871        68
Cristobal Butcher       2969        77
Billy Ennals            2191        82
Clarisa Freeze          2135        70
Dante Galentine         1194        89
Nia Stutes              2872        97
Suzi Tait               2519        82
Ciera Woolery           1531        81
```

```
sdb: LI

Student Record Database sorted by Student ID

Dante Galentine          1194        89
Ciera Woolery            1531        81
Clarisa Freeze           2135        70
Billy Ennals             2191        82
Suzi Tait                2519        82
Dorethea Benes           2583        97
Teisha Britto            2871        68
Nia Stutes               2872        97
Cristobal Butcher        2969        77

sdb: FN
Enter name to search:  Freeze

Student Name:       Clarisa Freeze
Student ID:         2135
Total Grade:        70

sdb: FI
Enter ID to search: 2519

Student Name:       Suzi Tait
Student ID:         2519
Total Grade:        82

sdb: foo
Error, invalid command.

sdb: FN
Enter name to search: Fubar
There is no student with that name.

sdb: FI
Enter ID to search: 0
There is no student with that ID.

sdb: QUIT
Program terminated…
```

Approach:

From the class notes and your prior experience with Java, you should be able to create the necessary B-Tree functions. It is strongly suggested that you code and test these separately. Once you have this worked out, you can use the UpperCasify example from the notes to figure out how to read in data sequentially. The last part requires writing a simple command interpreter which provides the user interface to the program. Again, write and test these functions separately before attempting to put the program together.

Instructions:

Write a "C" program, initiated from the command line, to implement the database described in the previous pages. It should be able to validate the correct number of arguments, and fail gracefully if the specified files cannot be found. Remember to make searches case insensitive.

To obtain full marks, your program must work correctly, avoid the use of arrays except for representing character strings, and be reasonably well commented.

Run the examples shown above and save your output to a file called database.txt. Place all of your source code in a single file, database.c

Upload your files to myCourses as indicated.

About Coding Assignments

We encourage students to work together and exchange ideas. However, when it comes to finally sitting down to write your code, this must be done *independently*. Detecting software plagiarism is pretty much automated these days with systems such as MOSS.

https://www.quora.com/How-does-MOSS-Measure-Of-Software-Similarity-Stanford-detect-plagiarism

Please make sure your work is your own. If you are having trouble, the Faculty provides a free tutoring service to help you along. You can also contact the course instructor or the tutor during office hours. There are also numerous online resources – Google is your friend. The point isn't simply to get the assignment out of the way, but to actually learn something in doing.

fpf/November 19, 2018.