Practice S12P08: Database II: Database Operations

http://www.comp.nus.edu.sg/~cs1010/4_misc/practice.html

Week of release: Week 12

Objective: Structures, searching and sorting

Task statement:

In today's world, many applications use database technology to manage a large amount of data in an organized way.

For example, a book club may store data about its members, books, and information about which member has read which book. A summary of such data can be generated and used for many practical purposes, such as to recommend books to a member based on other members who have read similar books.

In this exercise, you are to write a program to read in some data, store them into arrays of structure variables, and perform some operations on the data to generate a summary.

More specifically, you are given the following structure type definitions:

```
typedef struct {
     int id;
     char name[MAX_LENGTH+1];
} member_t;
```

```
typedef struct {
     int memID;
     char title[MAX_LENGTH+1];
} record_t;
```

The variables of the structure type **member_t** are used to store the ids and the names of the members. For example, {111, "Mary"} refers to a member whose id is 111 and name is Mary.

In addition, the variables of the structure type **record_t** are used to store who has read which book. For example, {111, "GameOfThrones"} refers to a member whose id is 111 and this member has read a book called "GameOfThrones".

The input to your program consists of multiple parts as shown on the right:

- The number of members
- The id and the name of each member
- The number of records
- The member ID and the title of the book read by this member.
- An integer (1 or 2) which represents the operation to be performed to generate a summary.

```
3
111 Mary
222 Peter
333 Lucy
6
333 MaryPoppins
222 WarAndPeace
111 GameOfThrones
222 HarryPorter
222 AloneInTheDark
333 TheWizardOfOz
1
```

Given these inputs, your program should perform an **Inner Join** (if the operation specified is 1) or a **Left Join** (if the operation specified is 2).

The effects of the two options are as described below:

- Inner Join: Every record is combined with a member if they share the same member ID.

  For example, after performing an inner join on the sample data, the summary should contain the following 6 results:
  - {"Lucy", "MaryPoppins"}
  - {"Lucy", "TheWizardOfOz"}
  - {"Mary", "GameOfThrones"}
  - {"Peter", "AloneInTheDark"}
  - {"Peter", "HarryPorter"}
  - {"Peter", "WarAndPeace"}

  ```
  typedef struct {
       char name[MAX_LENGTH+1];
       char title[MAX_LENGTH+1];
  } result_t;
  ```

  These results are stored in structure variables of the structure type **result_t** as shown above on the right.

- Left Join: Every record is combined with a member if they share the same member ID. In addition, for each member who hsas not read any books (i.e., his/her ID does not appear in the records), the member is combined with a dummy record which does not refer to any concrete book.
  For example, if the record {"Mary", "GameOfThrones"} is removed from the sample data and a left join is performed, the summary should contain the following 6 results:
  - {"Lucy", "MaryPoppins"}
  - {"Lucy", "TheWizardOfOz"}
  - {"Mary", "***"}
  - {"Peter", "AloneInTheDark"}
  - {"Peter", "HarryPorter"}
  - {"Peter", "WarAndPeace"}

  Note that after removing the abovementioned record, Mary has not read any book and her id does not appear in the record. Therefore, her name is combined with a dummy record "***", which is the 3rd result in the list.

(You are encouraged to read up online for more information about these operations.)

Your program, **bookclub.c**, should contain the following three functions:

- **readInputs()**: This function reads in all the inputs as described above
- **innerJoin()**: This function takes in all the inputs and performs an inner join. It returns the results in an array, as well as the number of the results.
- **leftJoin()**:This function takes in all the inputs and performs a left join. It returns the results in an array, as well as the number of the results.

The **sortResults()** function and the **printResults()** function are given to you. You should use them as they are to sort the results (by member name and book title in alphabetical order) before printing the results.

You may assume that 1) the maximum length of a name or a book title is 50 (i.e., MAX_LENGTH as defined in the program), 2) a name or a book title only consists of English letter, 3) the maximum number of members is 20 (i.e., MAX_CUSTOMER), 4) the maximum number of reading records is 100 (i.e., MAX_RECORD), and 5) the maximum number of results generated is 1000 (i.e., MXA_RESULT).

You may also assume that 1) the IDs in **member_t** are the non-empty, unique identifiers for customers, and 2) the member IDs in the reading records always refer to an existing customer.

Sample runs

```
Enter number of members: 3
Enter members:
111 Mary
222 Peter
333 Lucy
Enter number of records: 6
Enter records:
333 MaryPoppins
222 WarAndPeace
111 GameOfThrones
222 HarryPorter
222 AloneInTheDark
333 TheWizardOfOz
Enter operation: 1
The results of the join are as follows:
Lucy MaryPoppins
Lucy TheWizardOfOz
Mary GameOfThrones
Peter AloneInTheDark
Peter HarryPorter
Peter WarAndPeace
```

(There is one more sample run on the next page)

```
Enter number of members: 3
Enter members:
111 Mary
222 Peter
333 Lucy
Enter number of records: 5
Enter records:
333 MaryPoppins
222 WarAndPeace
222 HarryPorter
222 AloneInTheDark
333 TheWizardOfOz
Enter operation: 2
The results of the join are as follows:
Lucy MaryPoppins
Lucy TheWizardOfOz
Mary ***
Peter AloneInTheDark
Peter HarryPorter
Peter WarAndPeace
```