Practice S12P08: Database I: Database Queries

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, an online shopping web application may store data about its customers, the items being sold, and the orders placed by the customers on the items. Such data can be queried for many practical purposes, such as identifying which customer has not been making purchases recently and find out which item is the top seller in the past week.

In this exercise, you are to write a program to read in some data, store them into arrays of structure variables and answer simple queries on the data.

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

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

```
typedef struct {
  int cusID;
  char category[MAX_LENGTH+1];
  int spending;
} record_t;
```

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

In addition, the variables of the structure type **record_t** are used to store how much a customer has spent in a particular category of products. For example, {111, "Cosmetics", 500} refers to a customer whose id is 111 and this customer has spent 500 dollars on Cosmetics.

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

- The number of customers
- The id and the name of each customer
- The number of spending records
- The customer ID, the category and the spending in each record
- A name

```
3
111 Mary
222 Peter
333 Lucy
15
111 Cosmetics 500
222 Cosmetics 300
111 Toys 400
(12 more records...)
Mary
```

Given these inputs, your program should 1) find the spending records of the customer with the given name, 2) sort the records in descending order of the spending, and 3) print only the category and the spending in these spending records.

For example, in the sample inputs, the given name is Mary. Your program should print Cosmetic 500 and Toys 400, among other spending records for Mary.

Note that the record for Cosmetics must be printed before the one for Toys since the spending for Cosmetic is higher (500 vs 400). Moreover, the second record in the list {222 Cosmetics, 300} is not printed since that record is for customer 222, whose name is Peter.

If 1) a customer of the given name cannot be found, or 2) a customer of the given name can be found but there is no spending record for that customer, your program should print a message should be printed to indicate that there is no record for the given name.

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

- **readInputs()**: This function reads in all the inputs as described above
- **findRecords()**: This function takes in all the inputs and finds the records for a particular customer. It returns these records in an array, as well as the number of records.
- **sortRecords()**: This function sort the records in descending order based on spending.

The **printRecords()** function is given to you. You should use this function as it is in your program.

You may assume that 1) the maximum length of a name or a category is 50 (i.e., MAX_LENGTH as defined in the program), 2) a name or a category only consists of English letter, 3) the maximum number of customers is 20 (i.e., MAX_CUSTOMER), and 4) the maximum number of spending records is 100 (i.e., MAX_RECORD)

You may also assume that 1) the IDs in **customer_t** are the non-empty, unique identifiers for customers (*i.e.*, *a primary key* of customers), 2) the names of the customers are non-empty and unique as well (*i.e.*, it is a *candidate key* of customers), and 3) the customer IDs in the spending records always refer to an existing customer (*i.e.*, a *foreign key* to customers).

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

Check the sample runs on the next page for input and output format.

Sample runs

```
Enter number of customers: 3
Enter customers:
111 Mary
222 Peter
333 Lucy
Enter number of records: 15
Enter records:
111 Cosmetics 500
222 Cosmetics 300
111 Toys 400
333 Bags 100
111 Books 300
222 Toys 600
222 Music 200
333 Cosmetics 200
111 Flowers 200
222 Electronics 500
111 Bags 100
222 Flowers 100
333 Electronics 300
333 Books 400
222 Shoes 400
Enter name: Mary
The records for Mary are as follows:
Cosmetics 500
Toys 400
Books 300
Flowers 200
Bags 100
```

(The outputs is as shown below if the last input in the same run above is changed to **John**)

```
No record can be found for John.
```