# CSI 402 – Systems Programming – Spring 2014

## Programming Assignment III

**Date given:** Mar. 4, 2014 **Due date:** Mar. 26, 2014
**Weightage:** 10%

The deadline for this assignment is **11 PM, Wednesday, March 26, 2014**. With lateness penalty, the program will be accepted until **11 PM, Friday, March 28, 2014**. The assignment *won't* be accepted after 11 PM, Friday, March 28, 2014.

Very important: Your source program must consist of two or more C files and you must also have a `makefile`. The C files (with extension "`.c`"), header files (with extension "`.h`") and the `makefile` must be submitted together using the `turnin-csi402` command. Instructions for using `turnin-csi402` and additional specifications for the `makefile` will be included in the `README` file for this assignment.

---

In this program, you are required to implement some query operations on a relational database. A discussion regarding relational databases and the operations to be implemented will be presented in class.

The executable version of your program must be named `p3`. It will be executed using a command line of the following form:

`p3` *configfile* *queryfile*

Here, the argument *configfile* represents the name of a text file (called the **configuration file**) that provides information about the relations stored in the database. The argument *queryfile* represents the name of another text file that contains queries to be processed by your program. Details regarding these files are given below.

**Specifications regarding relations:** The name of any relation or any attribute of a relation is a string of length at most 15 characters. The name of a relation or an attribute starts with an upper or lower case letter. The remaining characters in the name may be upper or lower case letters, digits or the underscore (`'_'`) character. The database will have at least one and most 10 relations. Each relation will have at least one and at most 10 attributes. Further, each relation will have at most 100 tuples. For each relation, there is a **schema file** (which is a text file) and a **data file** (which is a binary file). The name of the schema file for a relation is obtained by appending the string `".sch"` to the name of the relation. Similarly, the name of the data file for a relation is obtained by appending the string `".dat"` to the name of the relation. Thus, if there is a relation called `Students` in the database, the names of the schema and data files for the relation are `Students.sch` and `Students.dat` respectively.

**Details regarding the schema file:** The first line of the schema file for a relation contains the number of attributes of the relation (a positive integer). This line is followed by $m$ lines, where $m$

is the number of attributes for the relation. Each of these $m$ lines, which describes one attribute of the relation, contains three items in the following order: the name of the attribute, the type of the attribute (one character; it is either `'I'` denoting an integer or `'S'` denoting a string) and the length (a positive integer) of the attribute in bytes. You can assume that for an attribute of type `'S'`, the length specified includes the byte for the terminating `'\0'` character. The three items describing an attribute are separated by one or more spaces.

**Details regarding the data files:** Recall that the tuples of each relation in the database are stored in a binary file. For any relation, the schema file gives information about the attributes of the relation. From the schema file, the number of bytes $\ell$ needed to store each tuple of a relation can be computed as the sum of the lengths of all the attributes of the relation. Each tuple of the relation is stored in the binary file as one record of length $\ell$ bytes. (The value of $\ell$ may vary from relation to relation.)

**Details regarding the configuration file:** As mentioned above, the configuration file is a text file. The first line of this file contains an integer that represents the number of relations stored in the database. This line is followed by lines each of which gives the name of a relation in the database. (Thus, if the first line has the integer value $n$, then there will be exactly $n$ lines after the first line.)

*You may assume that the configuration, schema and data files do not contain any errors.* An example of a database containing two relations and the corresponding configuration and schema files are shown on pages 4 and 5 of this handout.

**Details regarding the query file:** The query file is also a text file. Each line of the query file contains a command to be executed. You may assume that each line of the query file has at most 100 characters, including the `'\n'` character. On each line, the command name and the arguments are separated by one or more spaces. The syntax of the commands and their meanings are discussed below. *Note that all the output obtained by processing commands, including errors identified in commands, must be written to* `stdout`.

(a) Command `nattr`: The syntax of this command is as follows:

`nattr` *relname*

In this command, *relname* specifies the name of a relation. In response to this command, your program must print the number of attributes of the specified relation.

If the relation specified in the `nattr` command is not in the database, your program should print a suitable error message to `stdout`.

(b) Command `tuplen`: The syntax of this command is as follows:

`tuplen` *relname*

In this command, *relname* specifies the name of a relation. In response to this command, your program must print the length (in bytes) of each tuple the specified relation.

If the relation specified in the `tuplen` command is not in the database, your program should print a suitable error message to `stdout`.

(c) Command `infattr`: The syntax of this command is as follows:

`infattr` *relname* *attrname*

In this command, *relname* specifies the name of a relation and *attrname* specifies the name of an attribute. In response to this command, your program should print the type of the specified attribute and its length (in bytes).

If the relation specified in the `infattr` command is not in the database or the attribute specified in the command is not part of the specified relation, your program should print a suitable error message to `stdout`.

(d) Command `count`: The syntax of this command is as follows:

`count` *relname*

In this command, *relname* specifies the name of a relation. In response to this command, your program should print the number of tuples in the specified relation.

If the relation specified in the `count` command is not in the database, your program should print a suitable error message to `stdout`.

(e) Command `project`: The syntax of this command is as follows:

`project` *relname* *attrname*

In this command, *relname* specifies the name of a relation and *attrname* specifies the name of an attribute. In response to this command, your program should compute the projection of the specified relation over the specified attribute and print the projected values. *Your program should eliminate duplicate values before printing the output to* `stdout`.

If the relation specified in the `project` command is not in the database or the attribute specified in the command is not part of the specified relation, your program should print a suitable error message to `stdout`.

(f) Command `select`: The syntax of this command is as follows:

`select` *relname* *attrname* *relop* *value*

As before, *relname* and *attrname* specify a relation and an attribute. The argument *relop* may be one of the following six relational operators: `==`, `!=`, `<=`, `>=`, `>` or `<`. The item *value* is either an integer (if the attribute type is integer) or a string (if the attribute type is string). While any of the above relational operators may be used when the attribute is of type integer, only the `==` and `!=` operators may be used when the attribute is of type string. In response to this command, your program should print each tuple of the relation that satisfies the specified condition.

Your program must produce a suitable error message to `stdout` for each of the following errors in a `select` command: (1) The specified relation is not in the database. (2) The specified attribute

is not part of the specified relation. (3) The value specified in the condition does not match the type of the attribute. (4) Relational operators other than `==` and `!=` are used with an attribute of type string.

(f) Command `quit`: The syntax of this command is as follows:

```
quit
```

When this command is encountered, your program should stop.

*You may assume that each command will be one of* `nattr`, `tuplen`, `infattr`, `count`, `project`, `select` *or* `quit` *and that each command will have the correct number of arguments.* Your program should *not* stop when an erroneous command is encountered. Instead, it should continue to process the remaining commands. Once the program starts processing commands, it should stop only when the `quit` command is encountered.

**Additional errors to be detected:** Errors to be detected while processing each command were discussed above. In addition, your program must also detect the following errors.

(i) Wrong number of parameters on the Unix command line to run the program. (The correct number of parameters is three.) This error message should be written to `stderr`, and the program must stop right away.

(ii) The program is unable to open the configuration file, the query file, a schema file or a data file. Here also, the error message should be written to `stderr`, and the program must stop right away.

**Information about `README` file:** The `README` file for this assignment will will be available by 10 PM on Sunday, March 9, 2014. The name of the file will be `prog3.README` and it will be in the directory `~csi402/public/prog3` on `acunix.albany.edu`.

---

## Examples of Relations, Configuration File and Schema Files

In the following example, we have two relations (called `Students` and `Courses`). Note that the names of the attributes appear as column headings in the tables. The binary files store only the actual tuples of the relation and not the column headings. (Thus, the first relation has five tuples and the second has four tuples.)

**Relation:** `Students`

| Name | Major | Minor | Totcr | Majcr |
|---|---|---|---|---|
| Smith,Robert | PSY | CSI | 57 | 39 |
| Woods,Jane | CSI | BUS | 97 | 68 |
| Ramsey,Elaine | BUS | PSY | 107 | 88 |
| Wharton,Tom | BUS | PSY | 117 | 98 |
| Baker,Norma | BIO | CSI | 39 | 25 |

**Relation:** `Courses`

| CName | CId | Instr | Credits |
|---|---|---|---|
| Systems Programming | CSI-402 | Ravi | 3 |
| Business Law | LAW-220 | Evans | 3 |
| Operations Research | MSI-430 | Kim | 4 |
| Independent Study | CSI-497 | Keller | 1 |

**The configuration file for the above relations:**

```
2
Students
Courses
```

**The schema file `Students.sch`:**

```
5
Name   S  25
Major  S  4
Minor  S  4
Totcr  I  4
Majcr  I  4
```

**The schema file `Courses.sch`:**

```
4
CName  S  25
CId    S  8
Instr  S  10
Credits I  4
```