Program 2 – Tables
Submission via BlackBoard
Due: 11/17/14 (Start of Class)

> **NOTE: All programs (C++) will be tested in the UNIX (cse machines) environment. All files will be loaded into a single directory/folder on the UNIX machines and will be compiled with the following command:**
>
> **g++ *.cpp**
>
> **The programs will be executed with the command:**
>
> **./a.out**

Implement the following using C++ classes (tuples and tables are separate classes). You are encouraged to use inheritance to share common code for both tables.

> **students** { id, name, last_name }   (primary key = "id")
>
> **grades** { id, class, grade }   (primary key = "id, class")

You will decide which data structure to use internally to store the tuples. For this assignment you are allowed to use any of the data structures you have implemented yourself or the C standard template library.

Your program must be able to **add tuples**, **remove tuples**, and **list the tables**.

Upon starting your program, it will display the prompt "**tables>** " (with a trailing blank space):

tables>

The user can then select between the following commands:
- students add
- students delete
- students display
- grades add
- grades delete
- grades display
- quit.

---
students add and **grades add**

Add must be followed by adequate parameters. The corresponding tuples will be added to the table. Then repeat the prompt.

tables> students add 123 josh smith
tables> students add 111 jim jones
tables> grades add 123 computers A
tables>

---
students delete and **grades delete**

Delete must be followed by adequate parameters. The corresponding tuples will be deleted from the tables. A tuple is deleted from a table via the primary key. If a tuple is deleted from the grades table, only a single tuple is affected. If a tuple is deleted from the students table, all grades entry for that student id must be deleted as well. Then repeat the prompt.

tables> students delete 123
tables> grades delete 111 biology
tables>

students display   and   grades display

List the tuples of the corresponding tables. Tuples are separated by hyphens and elements of the tuples are enclosed in parenthesis and separated by commas.
For the students table, sort the output by id. For the grades table, sort the output by id first and by class second. If there are no tuples in the table display "empty table"

```
tables> students display
(111,jim,jones)-(123,josh,smith)-(222,some,person)
tables> grades display
(111,computers,C)-(123,computers,A)-(123,math,B)
tables>
```

quit

This command does not take any arguments. It terminates the program.

```
bst> quit
```

You must make your program "fool-proof", i.e. implement error checking. Each error output must start with "Error!" followed by an appropriate message.

NORMAL PROGRAM EXECUTION TEST (40 pts):

```
tables> students add 123 josh smith
tables> students add 111 jim jones
tables> grades add 123 computers A
tables> grades add 123 biology C
tables> grades add 123 chemistry B
tables> grades display
(123,biology,C)-(123,chemistry,B)-(123,computers,A)
tables> students display
(111,jim,jones)-(123,josh,smith)
tables> students delete 123
tables> grades display
empty table
tables> students delete 111
tables> students display
empty table
tables> quit
```

ERROR CONDITON TESTING (40 pts)

```
tables> tsudents add 123 josh smith
tables> students add 111 jim
tables> grades dad 123 computers A
tables> grades add 123 biology C D
```

```
duplicate add student
duplicate add grade for a student
```

Your program will be judged on the following:

40% - Passes I/O requirements
40% - Code satisfies requirements of assignment
20% - Professional coding style
      - 5% Adequate comments
      - 5% Modularity (small main function, separate functions, etc)
      - 5% Readability (line length, indentation, variable names)
      - 5% Elegance (e.g. use   if (condition)   instead of   if (condition == true)

Your program will receive a 0 if it does not compile (g++ *.cpp) on the UNIX [cse] machines.