

## **Project #02: iClicker Analysis, part 2**

**Complete By:** Tuesday, September 26<sup>th</sup> @ 11:59pm

**Assignment:** C++ program to perform more clicker analysis

**Policy:** Individual work only, late work *\*is\** accepted (see “Policy” section on last page for more details)

**Submission:** online via repl.it (exercise P02)

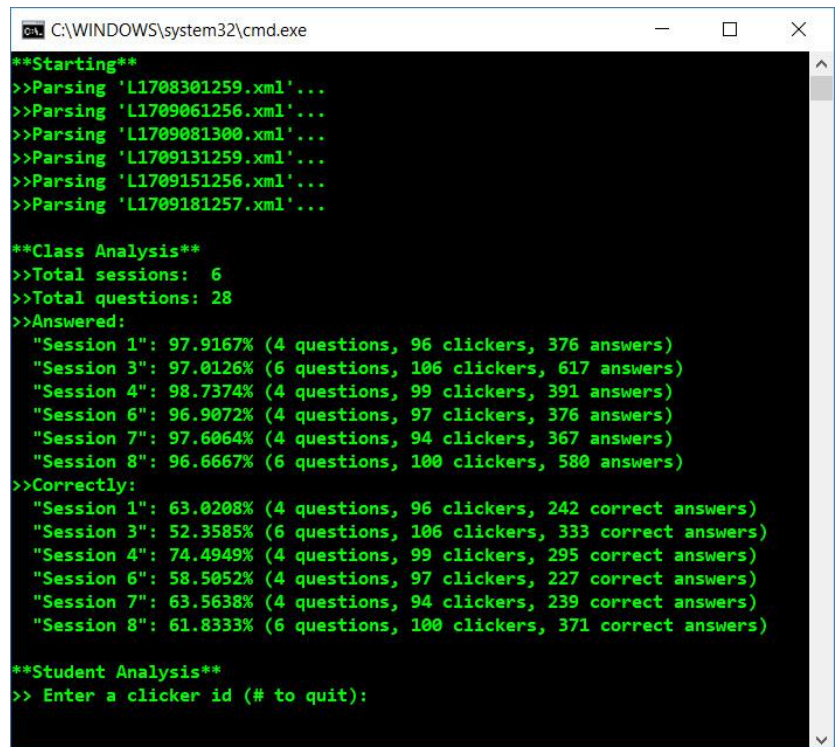
### Assignment

In the first project we analyzed a single clicker data file. In this assignment we’re going to write a C++ program to analyze N > 0 clicker files and perform a more detailed analysis. The program reads a file named “files.txt”, which contains the name of each clicker file to process (one per line).

Example:

```
L1708301259.xml  
L1709061256.xml  
L1709081300.xml  
L1709131259.xml  
L1709151256.xml  
L1709181257.xml
```

The program then parses the data and builds one or more data structures of your choice. The first part of the output is a class analysis shown to the right.



```
C:\WINDOWS\system32\cmd.exe  
  
**Starting**  
>>Parsing 'L1708301259.xml'...  
>>Parsing 'L1709061256.xml'...  
>>Parsing 'L1709081300.xml'...  
>>Parsing 'L1709131259.xml'...  
>>Parsing 'L1709151256.xml'...  
>>Parsing 'L1709181257.xml'...  
  
**Class Analysis**  
>>Total sessions: 6  
>>Total questions: 28  
>>Answered:  
  "Session 1": 97.9167% (4 questions, 96 clickers, 376 answers)  
  "Session 3": 97.0126% (6 questions, 106 clickers, 617 answers)  
  "Session 4": 98.7374% (4 questions, 99 clickers, 391 answers)  
  "Session 6": 96.9072% (4 questions, 97 clickers, 376 answers)  
  "Session 7": 97.6064% (4 questions, 94 clickers, 367 answers)  
  "Session 8": 96.6667% (6 questions, 100 clickers, 580 answers)  
>>Correctly:  
  "Session 1": 63.0208% (4 questions, 96 clickers, 242 correct answers)  
  "Session 3": 52.3585% (6 questions, 106 clickers, 333 correct answers)  
  "Session 4": 74.4949% (4 questions, 99 clickers, 295 correct answers)  
  "Session 6": 58.5052% (4 questions, 97 clickers, 227 correct answers)  
  "Session 7": 63.5638% (4 questions, 94 clickers, 239 correct answers)  
  "Session 8": 61.8333% (6 questions, 100 clickers, 371 correct answers)  
  
**Student Analysis**  
>> Enter a clicker id (# to quit):
```

The second part of the assignment is to then input clicker ids from the user, and for each clicker id, output an analysis of how that student performed across the different sessions. A screenshot is shown on the next page. This is repeated until the user enters “#”, at which point the program ends.

Your program must match this output exactly since it will be auto-graded using the repl.it system. Assume the master file is always named “files.txt”, and you may assume the referenced files exist (though it doesn’t hurt to check).

```
C:\WINDOWS\system32\cmd.exe

**Student Analysis**
>> Enter a clicker id (# to quit): 123456
** not found...

>> Enter a clicker id (# to quit): 990F32A4
"Session 1": 4 out of 4 answered, 75% correctly
"Session 3": 6 out of 6 answered, 100% correctly
"Session 4": 4 out of 4 answered, 100% correctly
"Session 6": 4 out of 4 answered, 75% correctly
"Session 7": 4 out of 4 answered, 100% correctly
"Session 8": 6 out of 6 answered, 66.6667% correctly

>> Enter a clicker id (# to quit): 8FFA2451
"Session 3": 5 out of 6 answered, 16.6667% correctly
"Session 6": 3 out of 4 answered, 0% correctly
"Session 7": 4 out of 4 answered, 50% correctly

>> Enter a clicker id (# to quit): A3E21E5F
"Session 1": 4 out of 4 answered, 100% correctly
"Session 3": 6 out of 6 answered, 100% correctly
"Session 6": 4 out of 4 answered, 100% correctly

>> Enter a clicker id (# to quit): A562F93E
"Session 4": 1 out of 4 answered, 0% correctly

>> Enter a clicker id (# to quit): blah
** not found...

>> Enter a clicker id (# to quit): #

**END**
```

## Input Files

The clicker files are XML files, in the same format as project #01. A set of input files can be downloaded from the course web site for testing purposes: see “Projects”, then “[project02-files](#)”. The files are being made available for each platform (Linux, Mac, and PC) so you may work on your platform of choice. NOTE: the best way to download the files is *\*not\** to click on them directly in dropbox, but instead use the “download” button in the upper-right corner of the dropbox web page.

## Our solution, or yours?

This project builds on project #01, so you should start from a working solution to that assignment. Build upon your own solution, or feel free to use ours. Our solution can be found on the course web page: see “Projects”, then “[project01-solution](#)”.

## Requirements

As is the case with all assignments in CS 341, how you solve the problem is just as important as simply getting the correct answers. You are required to solve the problem the “proper” way, which in this case means using modern C++ techniques: classes, objects, built-in generic algorithms and data structures, lambda expressions, etc. It’s too easy to fall back on existing habits to just “get the assignment done.”

In this assignment, your solution is required to do the following:

- As in project #01, use `ifstream` to work with files. Use only standard C++ to parse the XML, and use one or more classes to model the data (e.g. `Student`). When creating classes, follow good design principles: (1) all data members must be private, (2) the class contains a constructor to initialize the data members, and (3) getter/setter functions are used to access/update the data members.
- For data structures, use one or more of the built-in C++ containers: `vector<T>`, `map<T1, T2>`, etc. When appropriate, use the built-in algorithms such as `find_if` and `count_if`.
- No explicit pointers of any kind — no `char *`, no `NULL`, no C-style pointers
- No global variables whatsoever; use functions and parameter passing.

Writing code that is not modern C++ will result in a score of 0. Using a container other than `std::vector`? 0. No classes? 0. Using global variables? 0. Using a 3<sup>rd</sup>-party library such as Boost? 0.

For online documentation, I typically use the site <http://www.cplusplus.com/reference/>. Another way is to google with the prefix `std::`, which refers to the C++ standard library. Example: for information about the C++ string class, google “`std::string`”. For the vector class, google “`std::vector`”.

## Programming Environment

To facilitate testing, we’ll be using the `repl.it` system for submission and grading; see “**P02 More iClicker Analysis**”. You can work entirely within `repl.it`, or you can download the input files from the course web page and work in the C++ environment of your choice; Visual Studio is a good choice, but is not required for this assignment. Any environment with a modern (C++11 or newer) compiler will suffice; note that **bert** and **ernie** do not support modern C++, though **bertvm** does.

## Submission

The program is to be submitted via <http://repl.it>: see the exercise “**P02 More iClicker Analysis**”. The grading scheme at this point is 90% correctness, 10% style — we expect basic commenting, indentation, and readability. You should also be using functions, parameter passing, and good naming conventions.

## Policy

Late work *\*is\** accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. via Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *\*cannot\** work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

<http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf> .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at <http://www.uic.edu/depts/dos/studentconductprocess.shtml> .