**Submission of Your Work**

You need to prepare and submit ONE SINGLE MS Word document to Canvas (in your lab section) as LastName\_FirstName\_Labxy.doc. It must contain:

* Your NAME only on page 1
* For ***each*** question:
  + Specify the question number.
  + Source code. Copy/Paste your final source code. You must include standard “comment header” even if code is provided. *Do Not* paste a snippet of your source code, it must be copy/pasted.
  + Initial test plan. After reading the question requirements, but **before** beginning any coding, create the test case table, below, completed through column Expected Output. Include in your report.
  + Final test plan. Write your program then complete the **test table** with actual output results and include in your report *AFTER* your source code.
  + Output results. Paste in a snippet of output showing results for **every listed test case in your final test plan**, labeled with test case #

Test Table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |

* Add / delete rows from Test Table as necessary
* Modify column widths as necessary
* Test both valid and invalid input
* Test for every output expected
* If failure is an expected output and it happens then that test Passes
* Any test that fails means the program must be fixed so that it passes the test
  + Failing tests need a new test row, ie 1a, 1b, etc, showing corrections from original

Given:

struct clientData {

int accountNumber;

char lastName[ 16 ]; // c-style string, not string

char firstName[ 11 ];

float balance;

};

Create a random access file of 101 records named “credit.dat” using these code segments in a function to initialize the file to empty records:

ofstream creditFile( "credit.dat", ios::out);

clientData blankClient = { 0, "", "", 0.0 };  
  
for ( int i = 0; i <= 100; i++ )  
 creditFile.write( reinterpret\_cast<const char \*>( &blankClient ), sizeof( clientData ) );

creditFile.close();

<ostream> member function *write* - outputs a fixed number of bytes beginning at a specific location in memory to the specific stream (file).

The *write* function expects the first argument of type *const char \*,* hence the use of

*reinterpret\_cast <const char \*>* to convert the address of *blankClient* to *const char \*.*The second argumentof *write* is an integer specifying the number of bytes to be written.Thusthe use of *sizeof( clientData )*. Since size will never change it must be declared as a constant variable; sizeof must only appear once in your program: for the constant declaration.

The first entry in the file will be skipped so that record 1 is at position 1 and not position 0.

Write data into the file [minimum of 15 records, not in 15 consecutive account numbers] getting all data from the user. Truncate user input if it is too long for the data field.

Read data from the file. Loop, asking user for an account number to find (range 1 to 100, 0 to end input. error message if not a valid account number and try again). Print out all data fields for this account.

Update an account. Loop, asking user for an account number to update (error message if not a valid account number and try again) or 0 to quit (do several updates). Update firstname, lastname, or balance (but NOT accountNumber).

Print out all records that do not have accountNumber of 0, formatting output into columns (label columns).