CIS 130 - Intro to Programming - Spring 2007
Homework Assignment #9 - **INDIVIDUAL** assignment

Homework #9 DUE**: BEGINNING** of class, Wednesday, April 25, 2007

**Purpose:** get practice with complete C++ programs

Note that use of the design recipe is still required for all functions, including C++ functions! But, use **C++ types** in **contracts** for C++ functions, and use == for examples for non-void C++ functions. (describe output for a specific example for void C++ functions)

**1.** Remember **line_of_X** from HW #5? Write a C++ version of function **line_of_X** that takes the number of X's desired as its parameter, and prints to the screen that many X's, followed by a newline character. It should not return *anything*.

For example, **line_of_X(7);** would cause the following to be printed to the screen:
XXXXXXX

Create files **line_of_X.cpp** and **line_of_X.h**.
BUT: you cannot test this in ~st10/expr_play or ~st10/funct_play2 (I don't *think*), because it doesn't return anything. You need a main function for that. SO – proceed to #2.

**2.** Write a **main** function whose purpose is to test **line_of_X** --- it should be in a file named **test_line.cpp**, and should call the function **line_of_X** in **line_of_X.cpp/line_of_X.h.**

It should call **line_of_X** at least 3 times. When all is working fine, **line_of_X.cpp**, **line_of_X.h**, and **test_line.cpp** will be ready for submission.

**3.** Remember **box_of_X** from HW #5? Write a C++ version of function **box_of_X** that takes two parameters, the number of rows in the desired box and the number of X's per row, and prints a "box" of X' s to the screen with that many rows and that many X's per row, ending up with a newline character. It should not return *anything*, and it must appropriately call **line_of_X**.

For example, **box_of_X(3, 5);** would cause the following to be printed to the screen:
XXXXX
XXXXX
XXXXX

Create files **box_of_X.cpp** and b**ox_of_X.h**.
BUT: you also cannot test this in ~st10/expr_play or ~st10/funct_play2 (I don't *think*), because it doesn't return anything. You need a main function for that. SO – proceed to #4.

**4.** Write a **main** function whose purpose is to test **box_of_X** --- it should be in a file named **test_box.cpp**, and should call the function **box_of_X** in **box_of_X.h/box_of_X.cpp**.

This testing main should ask the user to type in the desired number of rows and X's per row, and then call **box_of_X** accordingly. (Be careful – be sure to include the files for ALL of the functions involved when creating the executable **test_box**.) When all is working fine, **box_of_X.h**, **box_of_X.cpp**, and **test_box.cpp** will b e ready for submission.

**5.** Remember **triangle** from HW #5? Write a C++ version of function **triangle** that takes one parameter, the

number of rows in the desired triangle, and prints a "triangle" of X's to the screen that is that many rows tall, with one X in the first row, two X's in the second row, and so on, ending up with a newline character. It should not return *anything*, and it must appropriately call **line_of_X**.

For example, **triangle(5);** would cause the following to be printed to the screen:

```
X
XX
XXX
XXXX
XXXXX
```

Create files **triangle.cpp** and **triangle.h**.
BUT: you also cannot test this in ~st10/expr_play or ~st10/funct_play2 (I don't *think*), because it doesn't return anything. You need a main function for that. SO – proceed to #6.

**6.** Write a **main** function whose purpose is to test **triangle ---** it should be in a file named **test_tri.cpp**, and should call the function **triangle** in **triangle.h/triangle.cpp**.

This testing main should continue to ask the user for triangle number of rows, each time then displaying a triangle of that many rows with the help of the **triangle** function, until the user enters a 0 or negative number of rows. (For full credit, you are expected to use the "classic" loop structure for this kind of loop.) (Be careful – be sure to include the files for ALL of the functions involved when creating the executable **test_tri**.) When all is working fine, **triangle.h**, **triangle.cpp**, and **test_tri.cpp** will be ready for submission.

**7.** Remember the C++ function **semester_grade.cpp/semester_grade.h** from HW #7? (It is available at the public course Moodle site, if yours isn't handy or quite correct.)

Write a **main** function whose purpose is to call s**emester_grade** appropriately for some number of students. It should be in a file named **class_grades.cpp** and should call the function **semester_grade** in **semester_grade.h/semester_grade.cpp**.

This main will ask the user how many students there are, and then ask for the homework, quiz, and final score for that many students, each time then using **semester_grade** to determine and then print to the screen the semester grade for that student.

Be careful – **semester_grade** is a "pure" function, and returns its result. How should it be called here? When all is working well, **class_grades.cpp** will be ready for submission.

When you are happy with these functions, you can either submit the .cpp and .h files mentioned above, OR you can use the following quickie-tool to build a file containing all of them (a tar file) and submit that one file instead (IF you have named your functions PRECISELY as given above...):

...if you are interesting in the quickie tool, then type the following at the cs-server prompt:
~st10/get_hw09

...give the name of a directory you want built, and when done, if all 10 files are listed on-screen as being in your new file, then you can submit the file whose name it tells you at the end.

(Note: you STILL use ~st10/130submit to submit this homework! But it is your choice if you submit the 10 .cpp and .h files in the usual way, OR use ~st10/get_hw09 and submit the single file it builds containing (hopefully) your 10 .cpp and .h files.)