COMP1200c – Assign 01
Due 11:59 pm – Thursday – January 29, 2015
Submit `assign01a.c` and `assign01b.c` via Canvas

Have you taken the syllabus quiz??

Your submitted file names(s) must be spelled and cased as instructed. The file extension must be correct, also.

PART A.

The following steps will guide you in entering the commands necessary to complete this assignment. These basic steps will be used in all future assignments. This assignment exposes you to several C statements that you will learn about in this course. For now, type as given. Try reading the programs to see if you can understand what they are doing.

0. Read Chapter 2 in the text. See the files in Resources on Canvas.

1. Start jGrasp.

2. We now want to create a new file and add the program below.

1.) Select ***File*** from ***Main Menu***, opens ***File Menu***
2.) Select ***New*** from the ***File Menu***
3.) A popup menu will open with several language choices, select ***C File***
4.) We want to give the file a name:
    i.) Select ***File*** from ***Main Menu***
    ii.) Select ***Save As*** from the ***File Menu***, a ***Save As*** window will open
    iii.) If in a lab, select the H: drive or your USB. Create a COMP1200 folder to help you organize your files.
    iv.) In the ***Filename*** text box, enter **`assign01a.c`,** then select ***Save***

Now you can begin typing in the program.

Where there are CAPS type the requested information. Enter the rest of the program below **exactly as is**, observing the column restrictions and including the comment and blank lines. The program prints a two line message welcoming you to COMP-1200.

Yes, you are to type the information between /* */ and include the blank lines.

```
/* TYPE YOUR NAME HERE
assign01a.c
COMP-1200 - TYPE CURRENT THE SEMESTER AND YEAR HERE
This a simple program which prints a message to
welcome you to COMP-1200.
*/
#include <stdio.h>
#include <math.h>

int main(void)
{
   printf("Welcome to COMP %d \n", 60 * (int)(sqrt(400.0)) );
   printf("\"War Eagle\"\n");

   return 0;
}
```

3. You are now ready to ***compile and link*** the program. First, click on the title bar of **`assign1a.c`** to make sure it is selected. On the ***Main Menu,*** select the ***Compiler***, then select ***Compile & Link***. In the message area at the bottom of the screen, you will see the results of the compile.

Look at the ***Results*** pane (at the bottom) to see if you have any errors. The most common error for this simple program is forgetting the terminating semicolon or not having matching parentheses or braces. Other errors can be introduced by typing errors. Make sure the program is entered exactly as shown above. When finished ***compile and link*** the program again. If you still have errors, proof read your program carefully.

4. You are now ready to run the program. From the ***Main Menu***, select ***Run*** then ***Run.*** The output of the program will be displayed in the lower window.

5. Now let's introduce some errors. Doing the following will not produce something to turn in, but it will give you an idea of how to deal with errors. Everyone will make errors. They can be a teaching tool.

- ☐ 16th line: change ***int*** to ***irnt***
- ☐ 18th line: change ***(int)*** to just ***int*** (remove parentheses)

- Note: ***int*** was previously ***blue*** but now ***black***. The editor is ***smart*** and can tell that they are now invalid C keywords (this doesn't help us with the 2nd error - it's not that smart!).
- Now compile and link the program again. Note the errors in the ***Results*** pane.

Errors are indicated with the line number of the error between two colons and a short description of the error. If you ***left-click*** the mouse on the error the cursor will move to the line where the error occurs in the edit pane (sometimes on longer programs you must scroll up one line to make the error appear). Try it. Fix this first error and compile again. Then follow the same procedure for the second error.

PART B:

1. Now it is time for you to design, code and run a larger program. The program we will write will compute and print the course grade, assignment grade and Banner grade according to the information in the COMP1200 syllabus.

To write this program, we will follow the five-step process for problem solving given in your book:

1. State the problem clearly.
2. Describe the input and output information.
3. Work the problem by hand for a simple set of data.
4. Develop a solution that is general in nature.
5. Test the solution with a variety of data sets.

Read through PART B **carefully**. This is your introduction to a development plan.

2. The first step of the five-step process should come easy in this course, since the problems that are assigned are generally stated clearly to begin with. In general, deciding exactly what the problem is and what you want the computer to do will be relatively difficult.

The problem we are going to solve is to compute a student's course grade, assignment grade and Banner grade according to the information in the COMP1200 syllabus.

3. The second step is to describe the input needed and the output given by the program. Like the first step, this step will be easier in this course then in the real world.

For our problem, the I/O description is the following. Note, some information is constant, i.e. the same for all students

Constant – the maximum number of i<clicker and syllabus quiz points, number of assignments,
Input –the student's exam grade, assignment grades, syllabus quiz points, and i<clicker points.
Output - the student's course grade, assignment grade and Banner grade.

There may be other information is needed to get the output.

Other – the average assignment grade, the grade percent, the extra credit percent

4. The third step is to work the problem by hand for a simple set of data. Let's try this for a case with the following information.

maximum i<clicker points	75
Number of assignments	10
exam01	86
exam02	94
final exam	92
assignments grades	0,0,95,92,69,54,61,62,70,68
syllabus quiz	10
i<click points	66;

The answers should be:

Course grade: 80.2
Assignment grade: 57.0
Banner grade: F

5. Now that we have a feel for how to solve the problem, we can do the fourth step: develop an algorithm. We will use top-down design meaning that we will start at the top of the problem and break it down into smaller problems that can be solved separately. The first technique of top-down design is problem decomposition - identification of the pieces of the problem that need to be solved sequentially.

DECOMPOSITION

1.) Get the student's assignment and exam grades
2.) Compute the assignment percent
3.) Compute the grade percent
4.) Compute the extra credit percent
5.) Compute the course grade
6.) Find the Banner letter grade
7.) Print the course grade, assignment grade, Banner letter grade

This problem could have been broken down differently. Also, a good program checks for bad input from the user. For simplicity, we will skip checking the input for errors, but as your programming skills improve, you should start considering where programs might go wrong and how you could design the program to recover from an error.

The second technique of top-down design is stepwise refinement - doing each step of the problem decomposition in enough detail that it can be easily converted to computer instructions. Flowcharts, pseudocode, or several other methods of specifying detail can be used to do the stepwise refinement. We will use pseudocode.

STEPWISE REFINEMENT

1.) Get the student's assignment and exam grades
2.) Compute the *assignment percent*
    Add each *assignment grade* to the *total*
    Divide *total* by *number of assignments*
3.) Compute the *grade percent*
4.) Compute the *extra credit percent*
5.) Compute the *course grade*
    Add *grade percent* and *extra credit percent*
6.) Find the *Banner letter grade*
    if *assignment percent* < 60.0 *letter grade* = F
    else if *grade* > 89.5 *letter grade* = A
    else if *grade* > 79.5 *letter grade* = B
    else if *grade* > 69.5 *letter grade* = C
    else if *grade* > 59.5 *letter grade* = D
    else *letter grade* = F
7.) Print the *course grade, assignment grade, Banner letter grade*

ALGORITHM

The DECOMPOSITION steps represent an algorithm that will be used to guide you as you write a program. Note, that this is computer language independent.

The DECOMPOSITION statements are used as comments in the program.

If needed, the sub-steps from the STEPWISE REFINEMENT can also be used as comments. Further STEPWISE REFINEMENT may be needed to break steps into sub-steps. As the STEPWISE REFINEMENT gets more detailed, the statements may look more like a computer language, but this is NOT a computer programming.

Note: items in **bold** are analogous to C statements or standard I/O library functions; items in *italics* are analogous to C variables (refer to program below to convince yourself)

6. Now that the algorithm has been designed, it is time to turn it into C code. Your C lectures have not yet covered all of the computer statements needed to solve this problem. But if you have followed the design process this far, you will be able to follow most of the code below. This is the second program you must type in and run for this assignment.

Now open the edit pane as described in **step A.2** above and enter the program found on the last page of these instructions.

7. Now repeat steps A.3 and A.4 to build and run your program. Use the new filename **`assign01b.c`** when you get to the Save As window. **This program is correct as it appears. However, you may need to correct any errors caused by typing errors**.

8. Now it is time for the fifth step in the five-step process for problem solving. The fifth step was to test your program with a variety of data sets. Start by testing with the two cases we worked out by hand in step 4. Your program results should match your hand-worked results. If they do not, correct the program and try again. Make up more test cases and test the program until you are sure it is correct. You want to try to test the program on all possible execution patterns.

9. When your program is saved, your work is stored in a directory created for you on the system. When you are logged onto the system, your directory appears to be a disk drive. The name of the drive is H:

10. You can get a printout of your program (make sure the program you want to print is in the edit pane) by selecting ***File*** from the ***Main Menu***, and then ***Print***. This will open the ***Print Window***, select ***OK*** and your source file should print to the printer indicated in the printer test box of the window) be aware that the pages you print are counted. The cost for printing in the lab is $0.06/per page.

11. Read the section **Standards for Documentation of C Programs**. Compare the guidelines given there to the program we developed in this lab.

12. Submit your `assign01a.c` and `assign01b.c` via Canvas.

13. If you are working in a lab, **BE SURE TO LOG OFF** of Microsoft Windows BEFORE YOU LEAVE THE LAB. This will log you out of the system. Failure to do so will mean someone else can sit down at the machine and **delete all the files in your directory or print lots of pages which will be charged to you.**

WARNINGS:

- **DO NOT LEAVE A USB IN THE PC**
- **DO NOT TURN OFF THE PC**
- **LOG OFF BEFORE LEAVING**

```c
// TYPE YOUR NAME HERE
// assign01b.c
// TYPE DATE DUE HERE
/*
  Compute the semester grade for COM1200
  assignments 40%, midterms 30%, final exam 30%,
  extra credit: syl quiz 1%, iclicker pts 2%
*/
#include <stdio.h>
// CONSTANTS
#define MAX_ICLICKER_POINTS 75
#define NUM_ASSIGNS         10

int main()
{
// INPUT - Get the student's assignment and exam grades
// Exam grades
   int exam01 = 86,
       exam02 = 94,
       final_exam = 92;
// Assignment grades
   int assignments[NUM_ASSIGNS] = { 0,0,95,92,69,54,61,62,70,68 };
// Extra credit
   int syllabusQuiz = 10,
       iclickPts    = 66;
// OUTPUT
   double grade,assignPerc;
   char   ltrGrade;
// OTHER variables
   int    a, assignTotal = 0;
   double gradePerc, extraPerc;

// COMPUTE
// Compute the assignment percent
// Get total grades
   for ( a=0;a<NUM_ASSIGNS;a++ )
   {
      assignTotal += assignments[a];
   }
   assignPerc = assignTotal / NUM_ASSIGNS;
// Compute the grade percent
   gradePerc = exam01 * .15 + exam02 * .15 + final_exam * .3 + assignPerc * .4;
// Compute the extra credit percent
   extraPerc = syllabusQuiz/10 + (double)iclickPts/MAX_ICLICKER_POINTS * .02 * 100;
   grade     = gradePerc + extraPerc;
// Find the Banner letter grade
   if ( assignPerc < 60.0 ) ltrGrade = 'F'; // greater than 60% required to pass
   else if ( grade > 89.5 ) ltrGrade = 'A';
   else if ( grade > 79.5 ) ltrGrade = 'B';
   else if ( grade > 69.5 ) ltrGrade = 'C';
   else if ( grade > 59.5 ) ltrGrade = 'D';
   else ltrGrade = 'F';

// OUTPUT - Print the course grade, assignment grade, Banner letter grade
   printf( "Course grade:     %4.1f \nAssignment grade: %4.1f \n", grade,assignPerc );
   printf( "Banner grade:     %c\n", ltrGrade );
   return 0;
}
```

Yes, you are to type the information between /* */ and include the blank lines.

Refer to Step 5 in PART A and correct your errors.