CMSC 106 Project #3 Fall 2011

Date assigned: Wednesday, October 19, 2011
Date due: Tuesday, November 1, 11:00 p.m.

1 Introduction

The purpose of this project is to become familiar compilation of multiple files, loops and parameter passing.

2 Project specifications

The submitted code should create output that is identical to what is shown in the public output files included in the `.tgz` file. But instead of looking at the whole project, you should work on implementing each function to do exactly what is described in the `game.h` file. I have already implemented the main for you as well as several public test files that will be discussed in lab on Wednesday, October 19. Each function needed has a prototype and a description in the `game.h` file. All of your code should be placed in the `game.c` file. Each function must do exactly what is described in the `game.h` description for that one function. DO NOT MODIFY ANY OF THE OTHER FILES YOU ARE GIVEN!

3 Step by Step Directions

1. Copy the file named `proj3.tgz` from the `projects` directory that is in your `106public` directory. Place your copy of this file into your `106` directory.

   ```
   cp ~/106public/projects/proj3.tgz ~/106
   ```

2. Change into your 106 directory.

   ```
   cd ~/106
   ```

3. Then extract the contents of the file.

   ```
   tar -xvzf proj3.tgz
   ```

4. Then, you will see that a new directory has been created. That new directory is in your `106` directory and is named Project3. Change into that directory.

   ```
   cd Project3
   ```

5. Then, edit the file named game.c. **DO NOT MAKE ANY CHANGES TO game.h OR TO main.c or to any of the other .c files you are given.**

6. Compile and run using one of the publicX.c files or the main.c file to provide the main function. Make sure to run your program testing it on several different input values.

7. Submit after you are confident that your functions are all written according to the specifications given in the functions.h file.

4 Development suggestions

Many people like to write their whole program, I have provided files containing main methods for you and you will be writing functions called by these main methods or possibly by other functions. The process of how to use the main function and the public source files as well as how to create your own testers will be discussed in the recitation section.

5 Project requirements

1. All of your projects for this course must have a comment near their top which contains your name, your Glue ID, your section number, your TA's name, and your university ID number.
2. All your C programs in this course should be written in ANSI C, which means they must compile and run correctly when compiled with the compiler `gcc`, with the options `-Wall`, `-Werror`, `-ansi` and `-pedantic-errors`.
3. You **may not** use any C language features except those introduced thus far in classs. Note that using any features of C other than these will result in **losing credit** when your project is graded. For this project you may not use arrays or structures.
4. The Campus Senate has adopted a policy asking students to include the following statement on each assignment in every course: "I pledge on my honor that I have not given or received any unauthorized assistance on this assignment." Consequently you're asked to include this pledge in a comment near the top of your program. See below for important information regarding violations of academic integrity.
5. Your program should be written using good programming style and formatting. Any instances of bad style will result in losing points. The good style issuse we will be grading for in this project consists of:
   - having an initial block comment with information as stated above.
   - use of adequate descriptive comments throughout, to indicate what your program is doing and how, also make sure there are not too many comments so that readability is hindered (points lost for both too many or too few)
   - neat and readable indentation and formatting, including consistent alignment of braces and indentation of subsidiary statements, and avoiding writing any statements or comments which are wider than the standard terminal window width of 80 columns,
   - writing clear and readable code,
   - not using any global (or file scope) variables,
   - having descriptive identifiers (variable names, function names, names for symbolic constants, etc) [note: using single letter variable names or names that are not descriptive of their purpose will result in loss of credit],
   - making sure that you have no "dead code" (code that can never be reached) and making sure that you have no "code duplication" (code that already appears another place in the program - or very similar code that could have been done once by creating a helper funciton,
   - use of symbolic constants for any special values which don't change and which are used in several places in your code,
   - avoiding disallowed C language features as discussed above,
   - and, I also suggest fully blocking all loops and conditionals – use a curly brace to make it more obvious what lines are dependent on which headers.

6 Submitting your project

1. Turn in your program using the "submit" program provided by running "submit". **This submits only the .c file containing your source code, not the executable version of your program**!

2. Your project must be electronically submitted by the date and time above to avoid losing credit. No projects more than three days late will be accepted for credit without prior permission and a valid, documented excuse, as described in the course syllabus.

   Note there is **no grace period** for project submissions (the submission deadline is enforced exactly at the time above, the 24–hour late deadline is enforced exactly 24 hours later, etc.) and exceptions cannot be made; see the course syllabus for details.

3. Only the version(s) of the project which you electronically submit can be graded; it is your responsibility to test the program you are submitting and verify that it works properly before submitting.

4. **Do not delay or wait until the last minute to submit your program.** If you do, and you have any trouble, it will be too late to come to office hours, and you'll lose credit on the project. But if you start early and have questions or run into any difficulty, you can come to the TAs' office hours for assistance.

   It is recommended that you complete and submit your project **at least one day prior** to the due date and then reread this project assignment to insure that you haven't missed anything important which could cause you to lose credit on your project. If you have, this will give you ample time to correct it.

7 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. **Any evidence** of impermissible cooperation on projects, use of disallowed materials or resources, or unauthorized use of computer accounts, **will be submitted** to the Student Honor Council, which could result in an XF for the course, or suspension or expulsion from the University. Be sure you understand what you are and what you are not permitted to do in regards to academic integrity when it comes to project assignments. These policies apply to all students, and the Student Honor Council does not consider lack of knowledge of the policies to be a defense for violating them. Full information is found in the course syllabus– please review it at this time.

8 Sample output

I have included public source files, public input files for the ones that need input and public output files. In class we will talk about how to use them. The output files given assume that input redirection was used on the executable created by compiling the corresponding public.c with the deterministic random seed and your game.c.

I also suggest you write similar small main functions to test each of the other functions that you implement.