CIS*2500: Assignment 1

Due January 16, 11:55 pm (git time)

The Problem

Write a program to play the game *obstruction* on a 6x6 grid. You can learn about the game here: `http://www.papg.com/show?2XMX`. There is no need to construct an AI player, you can simply have two human players use the same keyboard.

Musts: We won't grade it unless you meet this standard

- Compile, with no errors using -std=c99 -Wall -pedantic(on the raspberry pi (raspbian))
- Run and draw the board without crashing.

The Basics: The first 80%

You can get 80% on this assignment by just following the instructions. If you meet each of the expectations listed, you should get most or all of the 80 marks allocated to basic functionality.

Expectations for your Game

- The board will be no more than 100 x 100 in size.
- The game should start with the cursor positioned at the top left
- The game is personalized by using the players' names in prompts
- The program prints short, informative instructions for prompts
- The program ignores all input that is not the i,j,k,l, keys or the letter q (for quitting). Exceptions made for input that is needed for enhancements.
- The player is prevented from moving the cursor outside the board area

Expectations for your Code

- Your solution uses at least one function that you have written.
- Your solution properly uses the ncurses library
- Root assignment folder has src/ include/ and bin/ subdirectories that are used properly (see the ncurses example files on the website for an example) item Each .c file has the required header (see the policies document)
- Source code is properly indented
- Comments about algorithm logic, function parameters and return values purpose are present.
- Variable and function names are meaningful and are in camelCase

Things you will need to know

- How to use library functions.
- How to link to a library.
- How to write your own functions.
- How to use git to clone, commit, push
- How to write a simple makefile
- How to place files in folders

Enhancements: How to get the last 20%

The final 20% of the marks for this assignment are earned by showing that you have mastered the programming constructs and that you are an adept problem solver.

You do not need to do all of the following to get graded on the last 20% of the assignment, but you do need to demonstrate (verbally when we grade you) that you have a solid understanding of how to approach them all and you need to have done at least 50% of the work described by these extensions. Some of them are harder than others.

- Make an AI player
- Offer hints to the current player
- Make the UI colourful and usable. Be ready to talk about why your UI design is well suited to this program.
- Have an accessible, personalized help feature.
- Keep score. Make the score persistent so that it reloads on a subsequent play of the game.

Deliverables: Things you must hand in

1. (via moodle) Submit your c files with **ALL OF YOUR IDENTIFYING INFORMATION REMOVED** via moodle to the Assignment 1 link. You do not need to submit your README or makefile to moodle. Do this submission first as its deadline is less flexible.
2. (via git) Your entire submission must aslo be in your A1 repository in git including the README file and a makefile.

Grading

This assignment will be graded twice. Your peers will grade your assignment workshop-style using moodle and the TAs will grade your assignment in the regular way via appointments. The two grades will be averaged. Grades for each assessment element are shown below:

- Programming Style: Programming style will be determined by running cpplint on your submission and deducting one point for each error or warning. TAs will give no marks for programming style, they will only deduct marks for errors. The version of cpplint we will use is in the files folder on the course website. You may check your own code against it as you code.
- Program Operation(5) Compilation, makefile, running, and correct input/output are all checked in this section.
- Game Play(5) Does your game play according to the rules given? Does it correctly identify winners/losers? Is the UI easy to use?

- Programming Constructs(5) Did you use ncurses correctly? Did you make effective use of functions? Did you write your own functions?
- Submission Organization(5) Have you organized your submission correctly into subfolders? Do you have a readme and makefile? Are they correct? Do you have the academic integrity header? This is the only assignment that will have marks for organization- all future assignments will simply have deductions for failing to organize.
- Enhancements(5) TAs will look at the enhancements you did. It is best to document your enhancements in your readme file in case your assignment gets graded without you present. Peer grading does not include an assessment of the enhancements.