

# CGS 3460 Computer Programming Using C, Spring 2008

## Project Specification

Due Friday, May 2 2008, at 7:30am  
No late submissions.

In this project, you will write a simple 2-player (Freestyle) Gomoku game.

The game is played on a  $15 \times 15$  grid, with 2 players, each player having black and white pieces respectively. The players play alternately, and in each move a player can place one of his pieces on a grid location. The first player to get an unbroken row of five stones horizontally, vertically, or diagonally is the winner. See, for example, <http://www.steffengerlach.de/gomoku/index.html> to play gomoku online.

In your implementation you will use `x` instead of black, and `o` instead of white. The goal of the program is to facilitate a 2-player Gomoku game, along with the ability to save and load games.

In the first screen, the user will be presented with a menu, presenting options to play a new game (Choice 1), Load a game (Choice 2) or Exit (Choice 3). If the user enters 1, then the next screen shows the grid, and prompts Player 1 to make the move. The rows of the grid are marked from `a` through to `o`, and the columns from 1 through to 15. The user will enter the position as a concatenation of the row and column. For instance, if the player wants to place a piece in the third row and 12th column, she will enter `c12` (This is reminiscent of way moves are represented in chess). In the next screen, the grid is refreshed with the previous player's move and the other player is asked to make a move. If one of the players wins, the program should point this out, and ask the user to press the enter key to get back to the first menu. At any point in the game, if the user wishes to quit, she can enter `x`, and she is asked if the game must be saved, or not. If she chooses to save, the program asks for a filename, and saves it in it. If not, the program gets back to the first menu. Analogously, the Load game option in the first menu, asks the user for a filename and loads the game stored in the file, and continues from where it was left off. If the file is in the incorrect format, the program must report an error and get back to the first menu.

**Extra credit:** Also, have a 1 player mode, i.e. where a player can play against the computer. So, when the user chooses new game, prompt for a 1 player/2 player (choices 1/2). Use a simple strategy, where the computer always blocks the player's longest unbroken chain so far.

Between screens, you may use the following function to clear the screen:

```
void clrscr(void)
{
    printf("\033[2J");/* Clear the entire screen.*/
    printf("\033[0;0f");/* Move cursor to the top left hand corner */
}
```

The file you save in (and load from) must have the following format:

```
<Mode>
<Next>
<blank_line>
```

<Row 1>  
...  
<Row 15>

<Mode> records the mode of the game. 0:if player1 vs player2, 1:player1 vs computer  
<Next> records who should move first when the game resumes. 0:player1 moves first, 1:player2 or the computer moves first.

<Row y> records all the stones in row y on the board. It consists of 15 characters. '0' represents a empty grid, 'x' and 'o' means this grid is occupied by a black stone or white stone respectively.

A sample save file looks like this:

```
1
0

0000000000000000
0000000000000000
0000000000000000
0000000000000000
00000o0000000000
00000ox000000000
000000x000000000
0000000x00000000
00000000o0000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```

The above file represents a game between player1 and the computer. When the game resumes, player 1 (black) will play first.

## Grading

The basic functionality of displaying the grid and moves correctly carries 40% of credit. Correctly recognizing and reporting when the game is over carries 20% of credit. 30% of the credit is for correctly saving to file and loading from file. Documentation carries 10% of credit.

## Collaboration

This project must be done individually by each student. While you may discuss broad ideas with other students, you may not see or copy someone else's code. We may perform pairwise comparison of your code using sophisticated tools with everyone else's and with other sources on the internet to detect plagiarism.

## Submission instructions

All submissions must be online through the Courseworx system. Your code must be in `proj.c`. In addition, include documentation in `README.txt` briefly explaining the data structures you use, and what each of the functions do. Also, list any known bugs in `BUGS.txt`. You will lose less credit for bugs you have identified yourself. Tar all these into `project.tar` and submit to Courseworx.