project[9]: tic-tac-toe and revising your old code

Due Thursday, 4/21/2016, 12:59:59pm

Project Goals

The goals of this project are to:

1. Build upon some prior projects
2. Utilize functions, multidimensional arrays
3. Play a game
4. Learn more about images

Important Notes:

1. **Formatting:** Make sure that you follow the precise recommendations for the output content and formatting. Your assignment will be auto-graded and any change in formatting will result in a reduced grade.

2. **Comments:** Header comments are required on all files and recommended for the rest of the course. Points will be deducted if no header comments are included.

Background

Please save your code in a file named: tictacimage.c. In this assignment, we will develop a tic-tac-toe game. But this time, the tic-tac-toe game should be 4x4. The rules are mostly the same, but you need 4-in-a-row to win. The game should follow the same rules and implement the same functions as for project[6] with the following exceptions:

- `check_three_in_a_row` should now be `check_four_in_a_row,` per the rules of 4x4 tic-tac-toe
- When a user enters a move (legal or illegal) the program will not automatically print out the state of the board. The board will only be printed when the user enters the ‘s’ command (described below).

The prompt should ask a user to enter a command, as so:

```
Enter a command for Player 1 ([row,col], c, s, p):
```

The legal commands include one of the following:

- a legal move
- c: clear the board

- s: print the state of the board as displayed in Figure 1 (as a set of characters, dashes for horizontal lines, pipes for vertical lines, plus signs where the lines cross, and x's and o's for the filled in spaces; or

```
Column:1    2    3    4
Row:     ____ ____ ____ ____
        |    |    |    |    |
1       |    |    | O  |    |
        |____|____|____|____|
        |    |    |    |    |
2       |    | X  |    |    |
        |____|____|____|____|
        |    |    |    |    |
3       |    |    |    |    |
        |____|____|____|____|
        |    |    |    |    |
4       |    |    |    |    |
        |____|____|____|____|
```

Figure 1: state of the board

- p: print the state of the board as an image (see below for more info)

For this assignment, we will be using the same image format from project[5], review that project description for more info on images. For more information on image file formats, see this article on Wikipedia (https://en.wikipedia.org/wiki/Image_file_formats)

Your program should add the following functions to the ones originally required for project[6]:

- clear_table: this function should take as an argument an array used to store the state of the tic-tac-toe board. The function should then clear the moves from the board. After the function should run, the state of the board should be a blank tic-tac-toe board. This function should not return anything.
- display_image: this function should take as an argument an array used to store the state of the tic-tac-toe board. The function should then print the board as shown in Figure 2 (below). This function should not return anything.

display_image

This function should print a tic-tac-toe board with a square size of 5, and line thickness of 1. O's moves should show up as a black square, X's moves should show up as a grey square. So, if a user makes the moves as described above, the print_image should print:

```
P2
# image.pgm
23 23
255
255 255 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 0 0 0 0 0 0 255 255 255 255 255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
255 255 255 255 255 0 128 128 128 128 128 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 128 128 128 128 128 0 255 255 255 255 255 0 255 255 255 255 255
```

```
255 255 255 255 255 0 128 128 128 128 128 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 128 128 128 128 128 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 128 128 128 128 128 0 255 255 255 255 255 0 255 255 255 255 255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255 0 255 255 255 255 255
```

and this would be the resultant image:

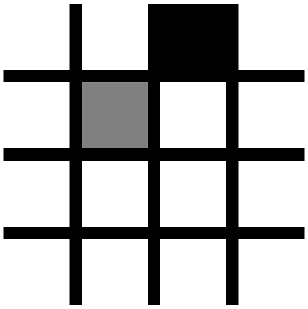

Figure 2

Challenge 1:

Use command-line arguments -s <int> and -t <int> to get a square size and line thickness for drawing custom-sized images. Please save you challenge as a separate file: tictacimage_c1.c

Challenge 2:

Write a simple AI to play tic-tac-toe, in this case, the computer will be player x. Please save you challenge as a separate file: tictacimage_c2.c

Submission details

The project needs to be submitted by Thursday, 4/21/2016, 12:59:59pm.

To submit your project, you will have to save your project files to an ECC machine using the Linux VM or the nomachine client:

- create a directory called "project9"
- save your *.c files in that directory
- save your description file into that directory
- DO THIS ONCE: Install the submission script ***<u>(don't type the '>' symbols)</u>***
  > cd ~

```
> wget http://www.cse.unr.edu/~newellz2/submit
> chmod +x ./submit
```

- TO Submit:

  ```
  > cd project9
  > ~/submit
  ```

The submission script copies all files in the current directory to our directory. You may submit as many times as you like before the deadline, we only keep the last submission.

Academic Honesty

Academic dishonesty is against university as well as the system community standards. Academic dishonesty includes, but is not limited to, the following:

Plagiarism: defined as submitting the language, ideas, thoughts or work of another as one's own; or assisting in the act of plagiarism by allowing one's work to be used in this fashion.

Cheating: defined as (1) obtaining or providing unauthorized information during an examination through verbal, visual or unauthorized use of books, notes, text and other materials; (2) obtaining or providing information concerning all or part of an examination prior to that examination; (3) taking an examination for another student, or arranging for another person to take an exam in one's place; (4) altering or changing test answers after submittal for grading, grades after grades have been awarded, or other academic records once these are official.

Cheating, plagiarism or otherwise obtaining grades under false pretenses” constitute academic dishonesty according to the code of this university. Academic dishonesty will not be tolerated and penalties can include canceling a student’s enrollment without a grade, giving an F for the course, or for the assignment. For more details, see the University of Nevada, Reno General Catalog.