CSCE 315 Team Project 3 Specification (Preliminary)

Dr. Daugherity

Due: 11:59 P.M. Monday, April 2, 2018

The project is to design and write a 3-D 4x4x4x4 Tic-Tac-Toe computer game program in C++17 with an ncurses interface. The human player tries to get four X’s in a row, column, or diagonal before the computer gets four O’s in a row, column, or diagonal, and *vice versa*. Note that because it’s 3-D there are lots of ways this can be done: don’t forget the body diagonals from one corner of the cube to the opposite corner!

The program must run on an 80x40 terminal window logged in to compute.cse.tamu.edu and

1. Start with a splash screen showing (at least) the name of the game, the team number and name, and the team members' names. Feel free to add an animation of a game being played (see “Extra Items” below). Have a highlighted START button that when the ENTER key is pressed the program explains how to play the game.
2. Display both the game (see below) and the top 5 scores (read in from the disk; the top scores file starts out empty).
3. Ask for the player's initials and display them with the player’s score read in from disk, or if none then display the initials with a score of 0 below the top 5 scores.
4. Instruct the player to use the arrow keys to select a cell and press ENTER. Place an X there, and check if the game is over (X wins, O wins, or the cat got the game). For illegal moves give an error message and instruct the player to try again.
5. If not, choose the best move to place the computer’s O (see below) and check if the game is over. If not, return to step 5. Do not take more than 10 seconds to decide on a move (see below).
6. When the game is over, if the player won then add one to the player’s score (number of wins), sort the list of 6 scores, and write the top 5 out to disk with initials. Then the next time the game is played, that file will be read in and displayed in step 2 above. Ask the player if they want to play another game or quit.

Draw the board using characters, perhaps something like this:

TOP PLANE BOTTOM PLANE

+--+--+--+--+ +--+--+--+--+ +--+--+--+--+ +--+--+--+--+

| 1| 2| 3| 4| |17|18|19|20| |33|34|35|36| |49|50|51|52|

+--+--+--+--+ +--+--+--+--+ +--+--+--+--+ +--+--+--+--+

| 5| 6| 7| 8| |21|22|23|24| |37|38|39|40| |53|54|55|56|

+--+--+--+--+ +--+--+--+--+ +--+--+--+--+ +--+--+--+--+

| 9|10|11|12| |25|26|27|28| |41|42|43|44| |57|58|59|60|

+--+--+--+--+ +--+--+--+--+ +--+--+--+--+ +--+--+--+--+

|13|14|15|16| |29|30|31|32| |45|46|47|48| |61|62|63|64|

+--+--+--+--+ +--+--+--+--+ +--+--+--+--+ +--+--+--+--+

and explain that cell 1 is above cell 17, etc. To debug the game you might ask the player to enter the cell number where they want to move, but the final version must use arrow keys and press ENTER to select a move: all user input and output **must** use **only** ncurses.

EXTRA ITEMS

* On the splash screen show a game being played. This could be a simple loop which pauses one second between moves and five seconds at the end.
* Allow the user to select X or O and play first or second.
* When there are four in a row (either X’s or O’s), make the background of those cells blink red.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This is a team project, with three or four students on a team. The instructor will assign the teams. (Note: If there are any problems with your team assignment, please talk to your TA promptly.) Choose a team leader who will coordinate the project.

Divide up the code, with each student on a team of 3 doing two of the six items on page 1; however, items 4 and 5 must be done by different people. If you have a fourth team member, that person must do one of the extra items.

A team of 3 will receive 5 points extra credit for doing one "Extra Item" or 10 points for doing two. A team of 4 will receive 5 points extra credit for doing a second "Extra Item" or 10 points for doing all three “Extra Items.”

Item 5 must use a minimax tree with a utility function such as

1000 \* (# of rows/columns/diagonals with 4 X’s

* # of rows/columns/diagonals with 4 O’s)

+ 100 \* (# of rows/columns/diagonals with 3 X’s and 1 blank

* # of rows/columns/diagonals with 3 O’s and 1 blank)

+ 10 \* (# of rows/columns/diagonals with 2 X’s and 2 blanks

* # of rows/columns/diagonals with 2 O’s and 2 blanks)

+ (# of rows/columns/diagonals with 1 X and 3 blanks

* # of rows/columns/diagonals with 1 O and 3 blanks)

You may change this formula if you have a better idea (such as preferring to move to a cell that blocks the most X lines and lies on the most unfilled O lines).

Note: The game tree has (potentially) 64! paths, which is greater than 1089, so you can easily run out of memory and/or time, since you must decide on a move in no more than 10 seconds. Some speedups to consider: pack the states into as few bits as possible and unpack as needed; experimentally determine how deep a breadth-first search can go without swapping to virtual memory and cut off the search at that depth; and/or have the minimax traversal generate the nodes as needed (with **new**) and recycle their memory (with **delete**) as soon as they are no longer needed.

Follow good style, and limit each function to no more than 24 lines (one terminal window). Each team member is expected to have a rough idea of how all the code works, and should be able to explain in detail how their own part of the code works.

Choose a clever name for your team (but keep it clean ).

Be creative in deciding how to meet these specifications in an attractive and user-friendly way, but get the basic functionality working before you try to make it too fancy, or you may run out of time!

Your program source code must be submitted to CSNET (team leader only). The project report (described below) should be submitted on paper to your TA, according to the outline below (one report per team). All team members will receive the same project grade, unless some team member does not do his/her part (see report outline below).

**Important! You must demonstrate your project to your TA or it will not be graded!**

Note about teamwork: Immediately exchange contact information with your teammates and schedule times to meet and work on the project outside of lab. As meeting scheduling can be difficult, use this lab time wisely! Attendance will be taken during lab, so that complaints of "We could never find a time to meet" will not be taken seriously.

REPORT OUTLINE

The project report must be printed on a laser printer. The report should include the following sections:

1. Team information (team name, members' names, who did what, did each member

do a fair share of the work)

2. Statement of the problem, significance, etc.

3. Restrictions and limitations

4. Explanation of your approach (analysis to choose a strategy for programming the project, how you coded it, etc.)

5. Sample run (screen shots)

6. Results and analysis, including post-production notes on changes you had to make to your design and why, difficulties, and solutions)

7. Conclusions - What did you show? What did you learn?

8. Future research (how your program could be improved or extended)

9. Instructions on how to run your program

10. Listing of the program, which must have adequate comments

11. Bibliography - references used, if any

**Deliverables and Requirements**

* All teams must use github.tamu.edu. Also give access to the TA and the instructor.
* Each team must maintain a development log (wiki page in github.tamu.edu titled "Development log") updated by the team members. This log will be graded. There is no designated format, except that you need to time stamp, write down the name, and write a brief description of the activity. We will check your daily progress.
* Major routines should include unit testing.
* Demo in the lab may be required.

1. Design documents: Follow the guidelines in [Scott' Hackett's "How to Write an Effective Design Document" (Writing for a Peer Developer)](http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document/). Include all four sections described in the guide.
   * Set up your design document ("Design document") as a wiki page in github.tamu.edu.
   * The design document should cover all 6 tasks plus any Extra Items you are doing.
   * Due 11:59 P.M. Wednesday, March 21, 2018.
2. Framework code: Upload working code implementing tasks 1, 2, 3, and 6. For other parts it is acceptable to print “Not implemented yet.”

Grading:

* + 10%: layout, style, comments
  + 20% each: tasks 1, 2, 3, and 6 work
  + 10%: development log
  + Due 11:59 P.M. Wednesday, March 28, 2018.

1. Final project code + report: The complete game program source code (to CSNET) and the report (on paper).
   * All tasks and any Extra Credit items work as specified.
   * All sections of the report are complete.
   * Individual work load distribution (percentage, must add up to 100%). Include this in section 1 of the report.
   * Development log (wiki page)
   * Formula for individual score calculation is the same as on Project 2.
   * Due 11:59 P.M. Monday, April 2, 2018.

**Submission**

* All submissions should be through csnet.cse.tamu.edu.
* Design doc submission should be a single PDF file uploaded to CSNET. This will be a printout of your wiki page.
* First, fork your latest project into an archival branch named: Submission 1 and Submission 2, for the two code submissions, respectively.
* Use the "Download ZIP" feature in github and upload the resulting zip file.
* As for the documents (development log, etc.), we will check the github project.
* Late penalty is 1% per 2 hours. So, if you're late 1 day, you lose 12%.