CS150A - Project 3: Game of Battleship Fall 2010: Riedesel & Conger

Here’s a final project which should be kind of fun: write a program to play a legal game of Battleship. If you’re unfamiliar with the game, there are a host of descriptions, resources and stories on-line. (For example, I learned that it was invented in the early 1900’s and was first sold as a pencil-and-paper game.) And there’s a fun little demo at <http://www.learn4good.com/games/board/battleship.htm> that should provide some inspiration. For starters, your program (“the enemy”)should perform these tasks:

* generate a 10x10 blank board and secretly organize his fleet on it
* receive your guesses (where to fire) and report the results
	+ (“hit”, “missed”, “sank submarine”, etc)
	+ if you enter invalid input, the program should respond gracefully, not crash
* report when the game is over, and report statistics (% of shots that hit something, etc)
* keep a datafile of top scores, with names of players and their scores – read this in at the beginning of play, store it in an array of struc, update it as necessary, and save it before ending.

You should make your game scalable. That is, make it easy to change the program to use a different-sized board, or more or fewer ships, etc. These constants can be loaded from a configuration file, or received interactively from the user, or just declared at the top with #define statements.

It’s difficult on our system to address particular locations on the screen, but you can use the *system(“clear”)* command to redraw the board at the same location after each shot, giving the illusion of a more interactive program. The system() prototype is found in the stdlib header file. If you’re getting into this, see if you can make your program shoot back at human player or a second computer player.

Collaboration:

Work with class members (including both 8:30 and 2:30 sections) on any or all aspects of the research and design. Form small teams (1-3 people) to actually code, test and run your program. It is essential to keep track of who did what, and where any useful information was found. Keep a log of the contributions of each member of the team, and use it to complete the table at the top of the grade sheet.

What and How to Submit your Solution:

On or before **Tuesday, Dec 7 at 11:59pm**, hand in (electronically\*) a file with your program source code and any supporting files (configuration file, top scores, etc). In class on December 8, hand in paper copies of the following:

* gradesheet (attached, print a copy) – fill in blanks for the handin account and filenames, and list all team members and the percent of effort they contributed to each phase of the project.
* cover page with program title, date, and names of all team members
* documentation of the subject research you did to prepare for this project (cite sources)
* explanation of steps taken to develop your solution, including any assumptions you made
* program source code (reread the Program Documentation Guidelines in Blackboard)
* explanation of how the program was tested and verified to work correctly
* conclusions: what did you learn from this project? What would you change about your approach to this problem if you had to start over?

 The report should be neatly formatted and typeset (eg: MS Word) with individual sections labeled. Use a single staple in the upper left (no covers). Submit a single report for each team.

**CS150A Programming Project 3:** **Battleship Game**  Dec. 7, 2010

|  |  |  |  |
| --- | --- | --- | --- |
| **handin account**  | **handin file name(s)** | handin date & time | total score |
|  |  |  |  |

|  |
| --- |
| *List Project Team members, and how much effort each member contributed to each part of the project* |
| **team member name (print)** | **plan/design** | **write code** | **test/debug** | **write report** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| total | 100% | 100% | 100% | 100% |

|  |  |
| --- | --- |
| **grading criterion** | **score** |
| Functionality (20 pts): Does it run as specified?* does the program compile?
* does it run (and does it stop normally)?
* does it play a legal game of Battleship
* does it display meaningful output on the monitor screen?
 |  |
| Features (20 pts): Does it have all required features?* does it report “miss”, ‘hit” and “sunk”
* does it reject and recover gracefully from erroneous input?
* does it store statistics at end of game
* is it scalable

extra credit (5 pts): exceptional programming effort or output formatting |  |
| Logical Design (20 pts): Is code efficient and well-organized?* use #define macros for problem parameters
* use functions to break problem into component parts
* design program flow to minimize wasted calculation effort
 |  |
| Internal Documentation (15pts): Comments, indenting, spacing, variable names?* header comments at top of file: identify authors and explain purpose and operation
* separately-labeled sections (prototypes, main and each function
* short comment identifying purpose of each function
* good indentation and spacing
* meaningful variable names
* appropriate comments
 |  |
| Testing (10pts): Was the program thoroughly tested?* report should explain testing methods used (what did you check during design, coding, scale-up and final solution to guarantee that you got the correct result?)
 |  |
| Report Presentation (15pts): Format and completeness of the accompanying report?* cover page: program name, date, student names, handin acct & filenames
* documentation of subject research and sources consulted
* explanation of solution development and any assumptions made
* example program output
* conclusions: what did you learn, what would you do differently next time?
* collaboration report (above)
 |  |