Project 2: A 2-Player Battleship Game

Posted: Tuesday March 6, 2007.
Due: 11:59PM, Wednesday March 28, 2007.

Project Objective:

1. master the process of completing a programming project.
2. master program selection (if, if-else, switch, etc.).
3. master command line argument and file I/O.
4. get familiar with formatted output.
5. explore and select data structures to store the input data (2-D array may help, but is not necessary).
6. learn how to handle interactive input data with simple user interface.
7. (function may help, but you don't have to use it.)

Project Description:

In this project, you will implement the game battleship for two people to play. Each player will secretly deploy 5 ships of different sizes into a 10x10 board. The 5 ships are: aircraft carrier (a) of size 5, battleship (b) of size 4, cruiser (c) of size 3, submarine (s) of size 3, and patrol boat (p) of size 2. Each ship will take that same number of consecutive squares on the board as its size, either horizontally or vertically. For example, a battleship will take 4 squares. The players will take turns to guess the location of the opponent's ships and shoot. A ship is sunk if all the squares it occupies are shot. The player who sinks all the opponent's ships first will win the game.

Your program should (1) read from two input files the two players' ship deployment and show them on the monitor; (2) ask each player to shoot and report whether a shot has hit any ship; (3) decide when the game is over and announce the winner; and (4) show each player's shot chart at the end of the game and output them in an output file. A master program, in the format of executable file, and sample input files will be posted on the class web site.

Input

Your program will read in input from two files, each consists of exactly 5 lines, where each line has exactly 4 characters (2 letters and 2 digits) separated by one space. Each line specifies the placement of one ship. For example, a deployment in the right table is described by the following file:

```
A 0 A 4
B 0 E 0
F 3 F 5
H 7 J 7
E 8 F 8
```

	0	1	2	3	4	5	6	7	8	9
A	a	a	a	a	a					
B	b									
C	b									
D	b									
E	b								p	
F				c	c	c			p	
G										
H								s		
I								s		
J								s		

Each square in the 10x10 board is represented by its row (A-J) and its column (0-9). Each line gives the two ends of the ship. It starts with the starting square of the ship, followed by one space, then the ending square of the ship. The starting square will be the left end of the ship if it is placed horizontally or the top end of the ship if it is place vertically. The five lines give the location of the 5 ships in the order of **a, b, c, s**, and **p**. The ships will not overlap, that is, any square can be occupied by at most one ship.

When the game starts, each play will take the shot by specifying the square to shot from the keyboard. A valid shot is a square on the board, a letter between A and J followed by a single digit between 0 and 9. For example, on A0, the starting end of the aircraft carrier will be hit; and shot I2 will miss.

Program Execution Flow

1. print out the board with player 1's ship deployment on the monitor (See the master program for exact output format.)
2. prompt player 1 for confirmation by printing out on the monitor:
   `Player 1: confirm the ship deployment (Y/N):`
3. on input `Y`, clear the monitor and repeat steps 1 and 2 for player 2.
4. terminate the program if any player enters `N`.
5. print out both players' shoot boards (not showing the locations of the ships): x for a hit, o for a miss, and – (minus sign) for squares have not been shot yet.
6. ask player 1 to shoot.
7. terminate the program if player 1 enters `Q` (for quit) as the row to shoot.
8. if player 1 enters an invalid square, ask the player to shoot again.
9. on a valid shoot selection, clear the monitor, tell player 1 whether the shot is a hit or miss
10. repeat steps 5-9 for player 2
11. repeat steps 5-10 for the next round of shoot until one player wins the game
12. clear the monitor and print out both players' shoot boards on the monitor and to an output file: x for hit, o for miss, - for square not being shot, and ship name (a,b,c,s,p) for square that the ship locates and not being shot.

Output

This project requires you write your output both to an output file and to the monitor.

Output file:

1. If a player confirms the ship deployment, write the following:
   `Player 1 confirms the ship deployment.`
   or `Player 2 confirms the ship deployment.`
2. If a player enters N on the ship deployment confirmation, write the following:
   `Player 1 fails to confirm the ship deployment.`
   or `Player 2 fails to confirm the ship deployment.`
3. If a player (say player 1) quits the game, write into the file each player's guess board (which consists of the player's shoots and the opponent's ship deployment. See step 12 in the program execution flow) and then the following sentence:
   `Player 1 quits the game.`

4. If a player (say player 1) wins the game, write into the file each player's guess board and then declare the player as the winner:

```
Player 1 wins!
```

Output to the monitor: see the master program on the web. Here is the list of the related printf() statements used in the master program for your convenience, although you don't have to use them exactly the way listed here:

```
printf("\nHIT!\n");
printf("\nMISS!\n");
printf("\n");
printf("Player 1's guessing board\n");
printf("Player 2's guessing board\n");
printf("player 1 turn to guess\n");
printf("player 2 turn to guess\n");
printf("please enter the row \n");
printf("please enter the column \n");
printf("The guess is invalid\n");
printf("Player 1: confirm the ship deployment (Y/N):");
printf("Player 2: confirm the ship deployment (Y/N):");
printf("Player 1 confirms the ship deployment.\n”);
printf("Player 2 confirms the ship deployment.\n”);
printf("Player 1 fails to confirm the ship deployment.\n”);
printf("Player 2 fails to confirm the ship deployment.\n”);
printf("Player 1 quits the game.\n”);
printf("Player 2 quits the game.\n”);
printf("Player 1 guess board\n");
printf("Player 2 guess board\n");
printf("o");
printf("x");
printf("\nplayer 1 wins!\n");
printf("\nplayer 2 wins!\n");
```

The command to clear up the monitor: `system("clear");`

Project Requirements:

1. You must program using C under GLUE UNIX system and name your program **p2.c**.
2. Submit your program **p2.c** electronically before the due time.
3. **IMPORTANT:** Your program's output, both to the output file and to the computer monitor, should be exactly the same as that produced by the master program.

Grading Criteria:

Correctness:	80%
Good coding style:	10%
Proper documentation:	10%
Late submission penalty:	-40% for the first 24 hours No submission will be accepted after the first 24 hours.