ECE 175 Computer Programming for Engineering Applications

Mid-Final Project: *Battleship*

Due Date: Wednesday November, 15th 2017 at 1PM, via D2L dropbox

Administrative details:

- Since the final project can be completed in a team of two students (maximum) or by yourself, **this mid-final project can be completed in a team. Sign up for your team (group of 2 or by yourself (work alone))** using the sign-up link below by 5 pm on Thursday November 9, 2017. After this date, you cannot change your team. https://docs.google.com/spreadsheets/d/1qphgzm5HRqTMWhrJFZTUzBC7uw2j_LqF4bkhP2An88A/edit?usp=sharing
- Submit your code (mid final.c) on D2L Dropbox by the due date and time above.
- **This mid-project assignment worth 10 out of 100 points** in your final project score. Late submission:
  - by 1 pm on Thursday November 16, 2017 (full score is 7 out of 10)
  - by 1 pm on Friday November 17, 2017 (full score is 4 out of 10)
  - 0 point for this part after 1 pm Friday Nov 17, 2017.
- Your mid-project code will be graded by ULA (no in-person grading for this part).

Note: The "complete" final project handout is posted in the other pdf file on D2L

Begin the code for Battleship:

For the final project you are required to use linked lists. The structure that you will be usng is

```
typedef struct ship {
  char shiptype[20];// e.g. Aircraft Carrier,
  int  shipsize;    // e.g.                 5,
  int  *X_Location; // Dynamically allocated array of length shipsize.
  int  *Y_Location; // Dynamically allocated array of length shipsize.
  int  *X_Hit;      // Dynamically allocated array of length shipsize.
  int  *Y_Hit;      // Dynamically allocated array of length shipsize.
  struct ship *pt;  // pointer to next ship on the list
} ship;
```

You are allowed to add attributes to this definition, but not remove any.

Here are some example ship types and sizes

Ship Type	Board Symbol	Ship Size
Aircraft Carrier	A	5
Battleship	B	4
Cruiser	C	3
Submarine	S	3
Patrol Boat	P	2
Kayak	K	1
Row boat	R	1

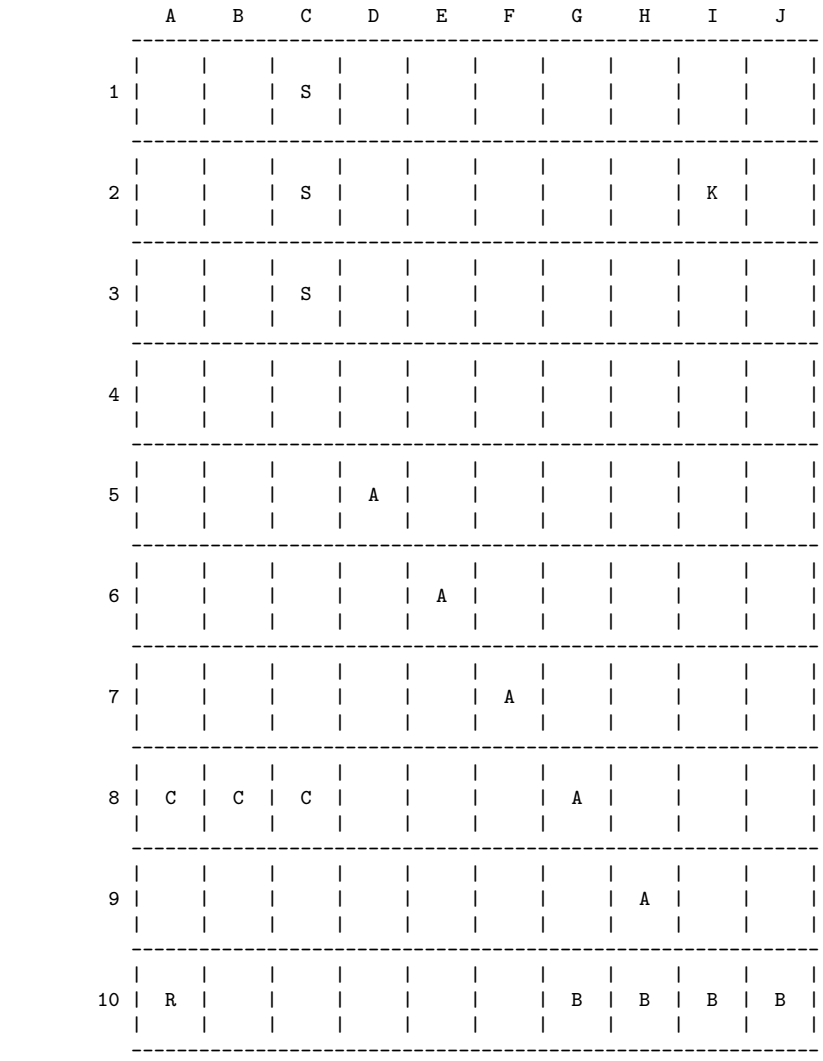
In preparation to code Battleship

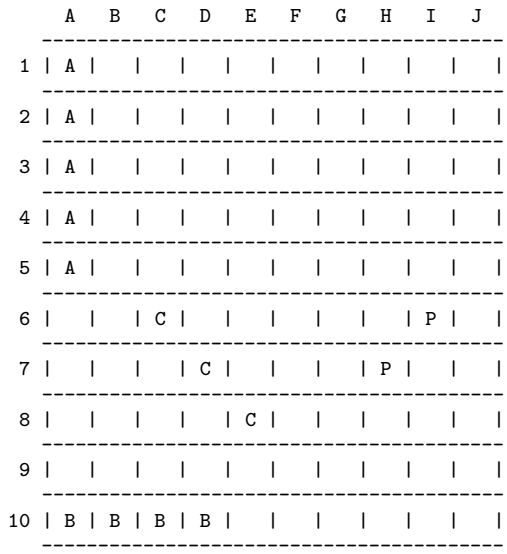
- Build two linked lists (one for each player) of different sizes: one linked list of 5 ships and another of 4 ships. To make the code easier for this assignment, do not use 2 of the same type of ship in either linked list (e.g. Player 1 list contains only 1 Aircraft Carrier).
- Ask the user to enter in the row and colum information for each linked list. Add the option to load the information using a text file named *Player1_Ships.txt* and *Player2_Ships.txt*. The first entry in the file is the number of ships to be read in. The file contains the ship type, the ship size, the row locations of the ship, and the column locations of the ship. Here is an example *Player1_Ships.txt* file.

```
6
Submarine
3
1 2 3
C C C
AircraftCarrier
5
5 6 7 8 9
D E F G H
Battleship
4
10 10 10 10
G  H  I  J
Kayak
1
2
I
RowBoat
1
10
A
Cruiser
3
8 8 8
A B C
```

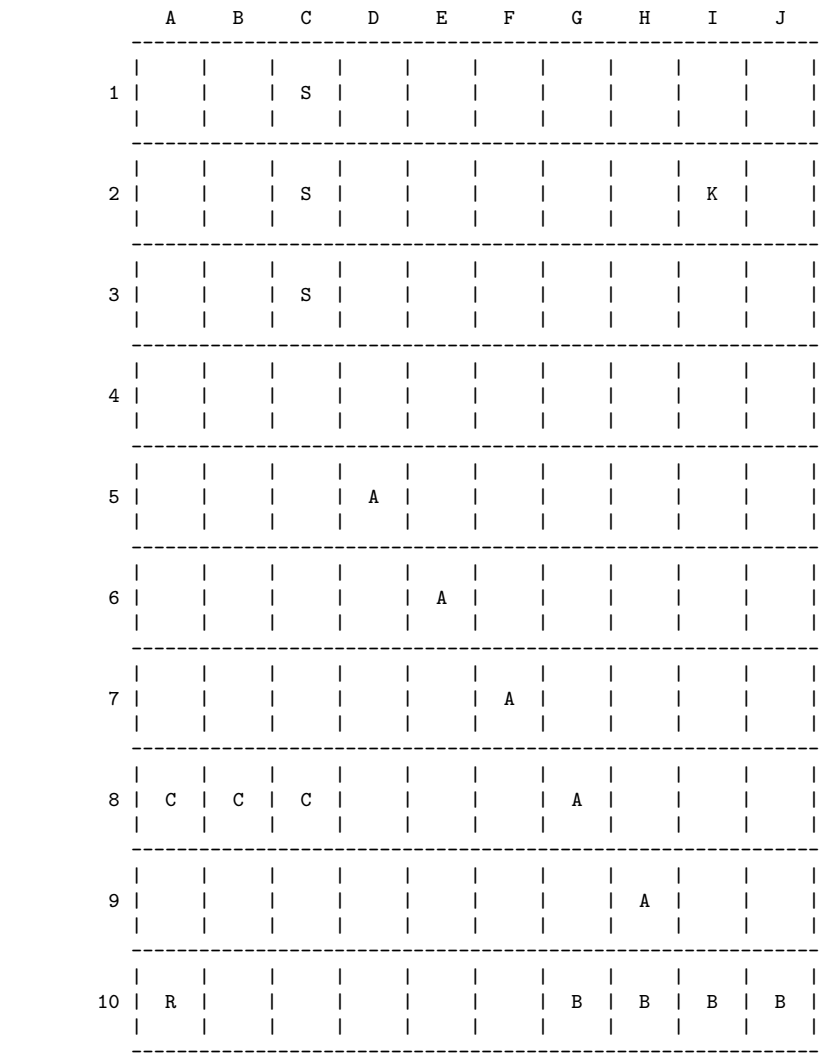
- Print the two grids and display the player's ships on each grid.
- Remove nodes of each linked list by asking the player which ship should be removed.
- Delete the ship node from the linked list and update the display. If no ship of that type exists then print an error message. If the ship type exists but is not part of the linked list then print this information to the screen.

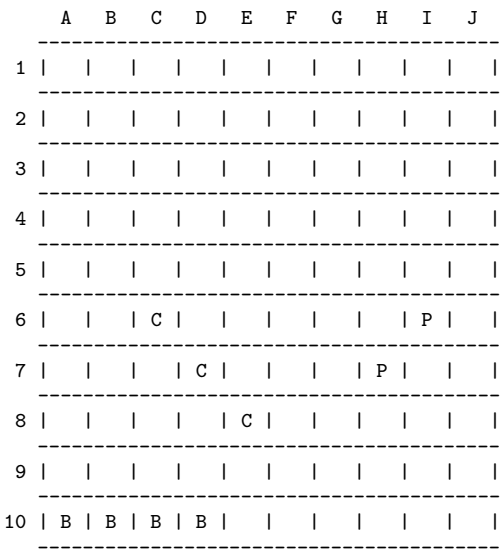
Example Situations: The sample execution below shows ships being removed from the grid. Red text indicates information entered by the user.



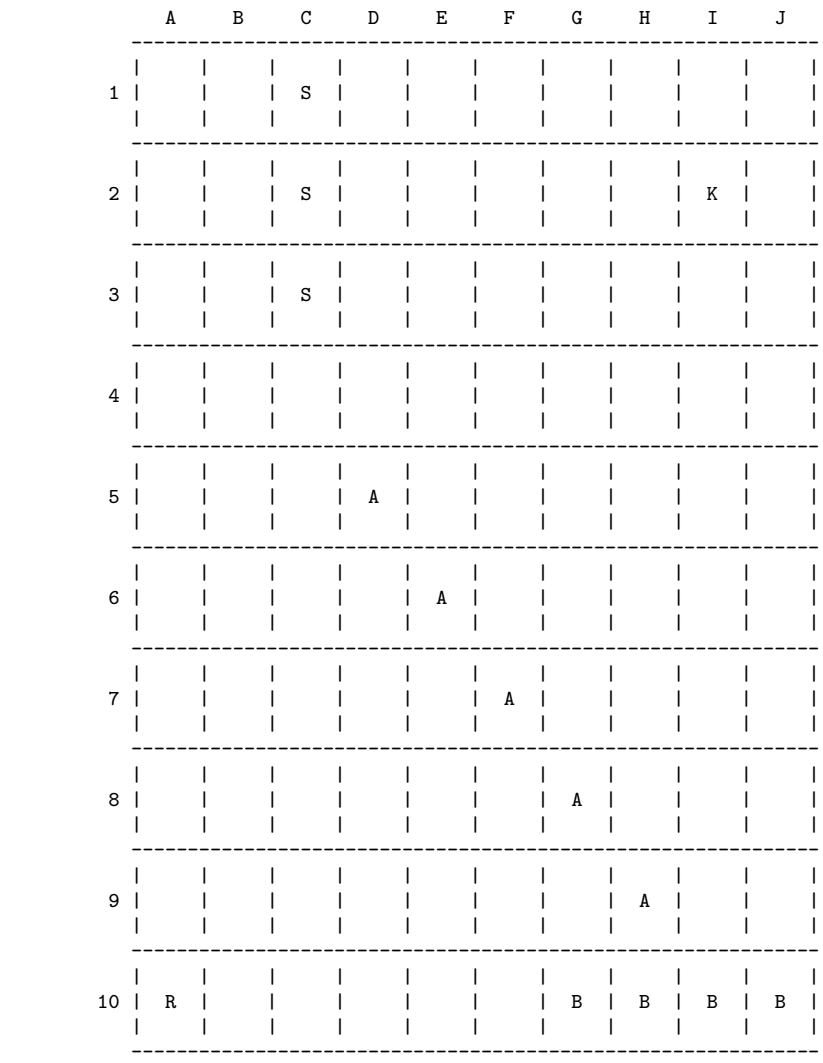


```
Player 1: What ship do you wish to remove from Player 2? aircraftcarrier
```





Player 2: What ship do you wish to remove from Player 1? patrolboat
Player 1 does not have a patrolboat on the board.
Player 2: What ship do you wish to remove from Player 1? Cruiser




```
Player 1: What ship do you wish to remove from Player 2?
```

Continue removing ships until the both boards are completely clear.