/\*

 \* File:   main.cpp

 \* Author: Scott Parker

 \* Created on April 10, 2017, 2:00 PM

 \* Purpose: Battleship game Project 1

 \* Notes: In the ship array 0 indicates ocean (cyan O), 1 indicates a ship (black X), 2 indicates

 \*     a ship that has been hit (red X) and -1 is a guess from the enemy player

 \*

 \*        In the the guess array 0 indicates ocean (cyan O), 1 indicates a miss (black X) and 2

 \*     indicates a hit (red X)

 \*/

//System Libraries

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

#include <cctype>

#include <fstream>

#include <string>

#include <cctype>

using namespace std;

//User Libraries

#include "colors.h"

#include "player.h"

//Constant to hold 2D array columns

//function to display the game board

//function to clear the gameboard

//Function to place a ship on the gameboard

//Function to see if ship is being placed in valid position

//Function to place all ships on the gameboard

//function to display the ship map only

//function to play the game

//function to reset all current game data

//Play a game with two human players

//Play a game versus the computer

//function to place all ships for computer player

//function for player to enter a guess

//function for computer to enter a guess

//Function to save the game

//Function to load a saved game

//Function to resume a saved game

//Function to resume a Human vs Human game

//Function to resume a Human vs Comp game

//Function to initiate player data and start game

//Executable code begins here! Always begins in Main

    //Set random seed

    //Declare Variables

//menu variable for choices

    //constant for number of rows

    //Player1 structure

    //Player2 structure

    //Game menu

    do{

        //Output switch menu screen

        //enter 1 to resume saved game

        //enter 2 for a new game

        //0 or unlisted number to exit

        //Loop to validate input

           //Resetting flags

           //ignore contents of buffer

        //keep requesting input until valid

        //Switch to determine the Problem

           //start a new game

           //Function to load a saved game

          //default option - exit menu

   //show menu while choices all active

    //Exit stage right! - This is the 'return 0' call end of main

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   newPlyr   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to initiate player data and start game

//\*\* Inputs: none

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    //player structure

    //loop through rows

        //loop through columns

           //set guess array value to 0

           //set ship array value to 0

    //return structure

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   resHum   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to resume a Human vs Human game

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //2 structures to track player hits and guesses

   //string to use for pausing screen

   //for loop to loop through rows

       //for loop to loop through columns

           //each non 0 occurance is a guess

               //increment total guesses

               //if guess value == 2 that guess was a hit

                   //increment number of hits

           //check if guess (value of array not zero)

               //increment total guesses if array value is a guess

               //check to see if guess was a hit

                   //increment hit counter if value was hit (array value 2)

   //begin do loop

       //output round number (guesses so far plus 1)

       //pause function to delay until <enter> pressed

       //for loop to output new lines to clear screen

       //call function to display player 1 game status

       //call function to enter guess and increment p1 hits if hit

       //increment total number of guesses so far

       //notify of pause

       //clear buffer

       //pause until enter

       //for loop to clear screen

       //notify of pause

       //pause until enter pressed

       //clear screen

       //call function to display game map for player 2

       //call function to guess for player 2 - increment hits if hit

       //increment player 2 number of guesses

       //notify of pause

       //clear buffer

       //pause until enter pressed

       //clear screen

       //output player 1 hits and guesses

       //output player 2 hits and guesses

       //notify of pause

       //clear buffer

       //pause until enter pressed

   //repeat loop until a player gets 14 hits or saves game

   //if game ends because of save then call save function

   //exit game message

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   resComp   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to resume a Human vs Comp game

//\*\* Inputs:  int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*          int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //string to be used for pausing

   //loop through rows

       //loop through columns

           //check if guess (value of array not zero)

               //increment total guesses if array value is a guess

               //check to see if guess was a hit

                   //increment hit counter if value was hit (array value 2)

           //check if guess (value of array not zero)

               //increment total guesses if array value is a guess

               //check to see if guess was a hit

                   //increment hit counter if value was hit (array value 2)

   //begin do loop

       //display player 1 game map

       //call guess function, increment player 1 hits if guess hits

       //increment number of player 1 guesses

       //call comp guess function increment player 2 hits if guess hits

       //increment player 2 guesses

       //output player 1 hits and guesses

       //output player 2 hits and guesses

       //notify of pause

       //clear buffer

       //pause until enter pressed or save char entered

   //loop until 14 hits scored by player or game saved

   //call game save function if necessary

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   resGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to resume a saved game

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //variable used for choice options

   //Load a saved game

   //choose computer or human opponent

   //force variable to upper case

   //loop to validate input type and range

       //output data range options

       //input choice

       //force to upper case

   //if choice is to play computer game

       //call function to resume game vs computer

   //otherwise

       //call function to resume game vs human

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   lodGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to load a saved game

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //string to hold file names from bones file. and hold names while counting lines

   //create filestream object

   //clear any game data in memory

   //open bones file in input mode

   //output error if file did not open correctly

       //loop through file line by line

           //get file line, increment to next line

           //output text from that line in file

   //close file

   //enter name of savefile to open

   //open binary file name provided

   //read in binary file data to populate game array data

       //read file to fill player 1 ship data

       //read file to fill player 1 guess data

       //read file to fill player 2 ship data

       //read file to fill player 2 guess data

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   savGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to save the game

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //hold file names from bones file and to hold names while counting lines

   //array of strings to hold file names

   //count variable and temp variable

   //create filestream object

   //open bones file in input mode

   //output error if file unable to open

   //otherwise

       //find the number of lines in the file

           //get file line, increment to next line

           //increment counter of the number of lines

   //close file

   //create dynamic array to hold bones file data

   //open bones file in input mode

   //output error if file fails to open

   //otherwise

       //loop through file line by line

           //get file line, increment to next array element

   //close file

   //loop through array line by line

       //display array element if not empty line

   //enter the name of new savefile

   //check for duplicate filename

        //add filename to end of array if no duplicate

        //overwrite if duplicate

    //loop through array

        //output contents of array

   //open bones file in output mode

   //output error message if file fails to open

   //otherwise

       //loop through array line by line

           //if line is not blank

               //output array element as line in file

   //close file

   //clean up memory and delete dynamic array

   //open binary file in output mode to save game data

   //write player 1 ship data to file

   //write player 1 guess data to file

   //write player 2 ship data to file

   //write player 2 guess data to file

   //close file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   entGues   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function for player to enter a guess

//\*\* Inputs: int pGuess[][COLS], int opShip[][COLS], int rows

//\*\* Outputs: 0 if miss, 1 if hit

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //character variable to hold column data for guess

   //variable to hold converted character variable as int

   //variable to hold row data for guess

   //variable to set to return 1 if hit, 0 if miss

   //bool variable set to false until successful guess entered

   //loop until successful guess

       //enter column

       //loop to validate that input was valid and in range

           //cheat and show enemy ships

           //activate cheat if out of range data input

           //input data for column to guess

       //if upper case data entered

           //subtract 65 from value and set column number guess to new value

        //else data is lower case character

           //subtract 97 from value and set column number guess to new value

       //input row number for guess

       //loop to validate input range and type

           //input row guess until valid entry

       //if guess array value not zero this value already guessed so repeat guess

       //otherwise if enemy ship array value = 1 HIT

           //set value of player guess array to 2

           //set value of enemy ship array to 2

           //set hit value to 1

           //set boolean as true since successfully entered guess

       //otherwise the guess was a miss

           //set player guess array value to 1 to show miss

           //set enemey ship array value to -1 to show miss

           //set hit value to 0 (since coordinate was a miss)

           //set boolean to true since guess was successfully made without error

   //return hit value from function

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   comGues   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function for computer to enter a guess

//\*\* Inputs: int pGuess[][COLS], int opShip[][COLS], int rows, int gues

//\*\* Outputs: 0 if miss, 1 if true

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //random column guess value

   //random row guess value

   //default hit value zero

   //boolean false until successful guess

       //if total guesses > 0 and guesses%11==0 then give comp a free hit

           //loop through rows while boolean false

                //loop through columns while boolean false

                   //if player has not guessed this coordinate already

                       //if this coordinate has an enemy ship that has not been hit

                           //set player guess value to 2 (hit)

                           //set opponent ship value to 2 (hit)

                           //set boolean to true

                           //set hit value to 1

       //otherwise if player has already guessed this position

           //get new column guess

           //get new row guess

       //otherwise if enemy has ship here

           //set player guess value to 2 (hit)

           //set enemy ship map value to 2 (hit)

           //set hit value to 1

           //set boolean to true (successfully entered guess)

       //otherwise position not guessed but no ship here

           //set player guess value to 1 (miss)

           //set enemy map value to -1 (to show enemy shot missed)

           //set hit value to zero

           //set boolean to true (guess attempted successfully)

   //return hit value

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   pGameH   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Play a game with two human players

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //string used for pauses

   //begin do loop

       //clear cin flags

       //notify of pause

       //pause until enter key pressed

       //clear screen

       //call function to display player 1 game data

       //call function for player 1 to enter a guess - increment hits if needed

       //increment player 1 guess count

       //notify of pause

       //clear screen

       //notify of pause

       //pause until enter key pressed

       //display player 2 game data

       //call function for player 2 to enter a guess - increment hits if needed

       //increment player 2 guess count

       //pause until enter key pressed

       //clear screen

       //output player 1 hits, player 1 guesses

       //output player 2 hits, player 2 guesses

       //pause until enter key pressed

   //loop until player has 14 hits or game saved

   //save game if required

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   pGameC   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Play a game versus the computer

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //string for pausing

   //begin do loop

       //call function for player 1 to enter a guess - increment hits if needed

       //increment player 1 guess count

       //call function for computer guess - increment hits if needed

       //increment player 2 guess count

       //call function to display player 1 game data

       //output player 1 hits and player 1 guesses

       //output player 2 hits and player 2 guesses

       //notify of pause

       //pause until enter pressed

   //loop until player has 14 hits or saves game

   //save game if necessary

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   pShipC   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to place all ships for computer player

//\*\* Inputs: int ship[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //variable to hold choice

   //variable to hold column number

   //variable to hold row number

   //boolean false until ship successfully placed

   //loop down from max ship size to min

       //output PT Boat if size = 2

       //output SUBMARINE if size = 3

       //output DESTROYER if size = 4

       //output BATTLESHIP if size = 5

       //begin do loop

           //random column value

           //random row value

           //randomly decide vertical or horizontal placement

           //if ship placement vertical

               //check to see ship will fit or overlap

                   //boolean false if ships overlap or out of bounds

               //otherwise

                   //Loop through rows from start position to ship size

                       //set array value to 1 (ship present)

                       //boolean true, ship successfully placed

           //otherwise ship placement horizontal

               //check to see if ship will fit or overlap

                   //set boolean false if ships overlap or out of bounds

               //otherwise

                   //loop through columns

                       //set array value to 1 (ship present)

                       //boolean true, ship successfully placed

       //loop until ship placement successful

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   newGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to play a new game

//\*\* Inputs:  int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*          int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //choice variable

   //pause input

   //Choose to play against a human or the computer

   //validate data

       //output available choices

       //input choice again until valid

   //if human opponent chosen

       //notify Player 1 to place ships

       //Place all ships on for player 1

       //pause until enter pressed

       //clear screen

       //notify Player 2 to place ships

       //pause until enter pressed

       //clear screen

       //call function to Play the game with two humans

   //otherwise

       //Place all ships on for player 1

       //place all ships for computer player

       //call function to Play the game vs. computer

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   clrData   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to reset all current game data

//\*\* Inputs: int p1ship[][COLS], int p1guess[][COLS], int p2ship[][COLS],

//\*\*         int p2guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //Reset game data for player 1

   //Reset game data for player 2

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   disShip   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to display a players ship data only

//\*\* Inputs: int ship[][COLS], int rows

//\*\* Outputs:  none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //loop through rows

       //output row number

       //loop through columns

           //if no ship present

               //output cyan 'O'

           //if undamaged ship present

               //output black 'X'

           //if damaged ship present

               //output red inverse 'H'

           //if other player shot this location but missed

               //output magenta 'o'

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   pAllShp   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to place all ships on the gameboard

//\*\* Inputs: int ships[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //loop down through size from Battleship to PT Boat

       //if size = 2 output PT Boat

       //if size = 3 output SUBMARINE

       //if size = 4 output DESTROYER

       //if size = 5 output BATTLESHIP

       //only output this message if error occurs

       //call function to place ship

       //call function to show ships map data

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   rngFind   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to check if the ship will fit on the board and is not

//\*\*           placed on top of another ship

//\*\* Inputs: int ships[][COLS], int rows, int size, int putRow,

//\*\*         int intCol, char verHor

//\*\* Outputs: true or false (boolean)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //boolean false if ship off board or overlapping another ship

   //if vertical placement

       //check to see if ship will be off the map

           //if ship off map set boolean to false

       //otherwise check overlap

           //loop through rows to check size

               //check to see if ship will overlap another

                   //if ships overlap set boolean to false

                   //set to true if ships overlap

                   //set boolean true if ship placement not overlap

   //otherwise placement horizontal

       //if ship will extend off the map

           //set boolean to false

       //otherwise check for overlap

           //loop through columns to check size

               //check to see if position will overlap

                   //set boolean to false ship placement not possible

                   //set overlap to true

                   //set boolean true if ship can be placed here

   //return ship placement boolean

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   putShip   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //Function to place a ship on the gameboard

//\*\* Inputs: int ships[][COLS], int rows, int size

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //variable to choose ship column placement

   //variable to choose horizontal or vertical placement

   //variable to hold char converted to int for column

   //variable to choose row placement

   //boolean set to false until ship successfully placed

   //loop while ship not placed

       //input column for ship placement

       //loop until range and type okay

           //input column for ship placement

       //if column choice is upper case letter

           //set int for conversion to value - 65

       //otherwise choice was lowercase

           //set int for conversion to value - 97

       //choose row to guess

       //loop check data type and range

           //input row until valid range and type

       //input vertical or horizontal placement

       //loop check data type and range

           //input vertical or horizontal until within range

       //if vertical placement

           //call function to check overlap and map ranges

               //if overlap or out of map set ship placemnt boolean to false

           //otherwise place the ship

               //loop through rows for size of ship

                   //set array value to 1 (ship present)

                   //set ship placement to true

       //otherwise placement is horizontal

           //call function to check overlap and map ranges

               //set ship placement to false if overlap or off map

           //otherwise place ship

               //loop through columns for ship size

                   //set array value to 1 (ship present)

                   //set ship placement to true

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   setGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to clear the gameboard and set all values to 0

//\*\* Inputs: int ship[][COLS], int guess[][COLS], int rows

//\*\* Outputs: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //run through the rows

       //run through the columns

           //set array for guesses all to false

           //set array for ships all to false

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   disGame   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//234567890123456789012345678901234567890123456789012345678901234567890123456789

//000000001111111111222222222233333333334444444444555555555566666666667777777777

//\*\* Purpose:  //function to display the game board

//\*\* Inputs: int guess[][COLS], int ship[][COLS], int rows

//\*\* Outputs:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   //output map legend

   //loop through rows

       //output row number

       //loop through columns

           //if guess array value zero

               //output cyan 'O'

           //if guess array value 1

               //output black 'M'

           //otherwise for array value 2 output red inverse 'H'

   //output map legend

   //loop through rows

       //output row number

       //loop through columns

           //if array value 0

               //output cyan 'O'

           //if array value 1

               //output black 'X'

           //if array value 2

               //output red inverse 'H'

           //otherwise for array value -1

               //output magenta 'o'