```
%%

warning('off','all') %Turns off all warning messages

clc, clear, format compact %Clean Start

 %Set up MAT-file variables

load Battleship

%Plot the board & label so user can see co-ordinates

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

 %Set up numerical arrays

X = {Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}};

%Make Cell w/ a bunch of 0s to make future logic easier.
%This cell is 12x12 so that any future arguments involving
%{Randi(1,10)+-1,Randi(1,10)+-1} are not undefined
%Outside barrier 3s what I like to call the "danger zone"
%As soon as computer defines arguments in this zone, computer will
%Automatically throw away the argument created

Z = cell(12,12);
N=2;
A=2;

while A<12

    Z{1,N} = 3;
    Z{12,N} = 3;
    Z{A,1} = 3;
    Z{A,12} = 3;
    Z{12,12} = 3;
    Z{1,12} = 3;
    Z{A,N}=0;
    N = N + 1;

    if N == 12

        N=1;
        A=A+1;

    end

end
%%

%Input coordinates of aircraft carrier and orientation


R = input ('What row do you want your aircraft carrier on? ');

C = input ('What column do you want your aircraft carrier on? ');

V = input ('What vertical orientation (Type 2 for up, 1 for down or 0 for none.) ');

H = input ('What horizontal orientation (Type 2 for left, 1 for right, or 0 for none.) ');
```

```
%Set up ship when orientation is right

if H == 1

% Place the left pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C+1} = Boat_Mid_hor;
Player_Board{R,C+2} = Boat_Mid_hor;
Player_Board{R,C+3} = Boat_Mid_hor;
% Place the right pointing end of the boat at position (2,3)
Player_Board {R, C+4} = Boat_FrontBack_4;

end

%Set up ship when orientation is left

if H == 2

% Place the right-pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C-1} = Boat_Mid_hor;
Player_Board{R,C-2} = Boat_Mid_hor;
Player_Board{R,C-3} = Boat_Mid_hor;
% Place the left-pointing end of the boat at position (2,3)
Player_Board {R, C-4} = Boat_FrontBack_3;

end

%Set up ship when orientation is up

if V == 2

% Place the downward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R-1,C} = Boat_Mid_vert;
Player_Board{R-2,C} = Boat_Mid_vert;
Player_Board{R-3,C} = Boat_Mid_vert;
% Place the upward pointing end of the boat at position (2,3)
Player_Board {R-4, C} = Boat_FrontBack_2;

end

%Set up ship when orientation is down

if V == 1

% Place the upward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R+1,C} = Boat_Mid_vert;
Player_Board{R+2,C} = Boat_Mid_vert;
Player_Board{R+3,C} = Boat_Mid_vert;
% Place the downward pointing end of the boat at position (2,3)
Player_Board {R+4, C} = Boat_FrontBack_1;

end

%Plot and label the board, and show Aircraft carrier

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)


subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)
```

```
ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)
%Clear Command Window

clc

%%

%Input coordinates of Battleship and orientation


R = input ('What row do you want your Battleship on? ');

C = input ('What column do you want your Battleship on? ');

V = input ('What vertical orientation (Type 2 for up, 1 for down or 0 for none.) ');

H = input ('What horizontal orientation (Type 2 for left, 1 for right, or 0 for none.) ');

%Set up ship when orientation is right

if H == 1

% Place the left pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C+1} = Boat_Mid_hor;
Player_Board{R,C+2} = Boat_Mid_hor;
% Place the right pointing end of the boat at position (2,3)
Player_Board {R, C+3} = Boat_FrontBack_4;

end

%Set up ship when orientation is left

if H == 2

% Place the right pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C-1} = Boat_Mid_hor;
Player_Board{R,C-2} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Player_Board {R, C-3} = Boat_FrontBack_3;

end

%Set up ship when orientation is up

if V == 2
% Place the downward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R-1,C} = Boat_Mid_vert;
Player_Board{R-2,C} = Boat_Mid_vert;
% Place the upward pointing end of the boat at position (2,3)
Player_Board {R-3, C} = Boat_FrontBack_2;

end

%Set up ship when orientation is down

if V == 1

% Place the upward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R+1,C} = Boat_Mid_vert;
Player_Board{R+2,C} = Boat_Mid_vert;
% Place the downward pointing end of the boat at position (2,3)
Player_Board {R+3, C} = Boat_FrontBack_1;

end

%Plot the board and show Battleship

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
```

```
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)
subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)
%Clear Command Window

clc

%%

%Input coordinates of Cruiser and orientation

R = input ('What row do you want your Cruiser on? ');

C = input ('What column do you want your Cruiser on? ');

V = input ('What vertical orientation (Type 2 for up, 1 for down or 0 for none.) ');

H = input ('What horizontal orientation (Type 2 for left, 1 for right, or 0 for none.) ');

%Set up ship when orientation is right

if H == 1

% Place the left pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C+1} = Boat_Mid_hor;
% Place the right pointing end of the boat at position (2,3)
Player_Board {R, C+2} = Boat_FrontBack_4;

end

%Set up ship when orientation is left

if H == 2

% Place the right pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C-1} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Player_Board {R, C-2} = Boat_FrontBack_3;

end

%Set up ship when orientation is up

if V == 2
% Place the downward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R-1,C} = Boat_Mid_vert;
% Place the upward pointing end of the boat at position (2,3)
Player_Board {R-2, C} = Boat_FrontBack_2;

end

%Set up ship when orientation is down

if V == 1

% Place the upward pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R+1,C} = Boat_Mid_vert;
% Place the downward pointing end of the boat at position (2,3)
Player_Board {R+2, C} = Boat_FrontBack_1;
```

```
end

%Plot the board and show Cruiser

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)


%Clear the Command window

clc

%%

%Input coordinates of Submarine and orientation

R = input ('What row do you want your Submarine on? ');

C = input ('What column do you want your Submarine on? ');

V = input ('What vertical orientation (Type 2 for up, 1 for down or 0 for none.) ');

H = input ('What horizontal orientation (Type 2 for left, 1 for right, or 0 for none.) ');

%Set up ship when orientation is right

if H == 1

% Place the left pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C+1} = Boat_Mid_hor;
% Place the right pointing end of the boat at position (2,3)
Player_Board {R, C+2} = Boat_FrontBack_4;

end

%Set up ship when orientation is left

if H == 2

% Place the right pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R,C-1} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Player_Board {R, C-2} = Boat_FrontBack_3;

end

%Set up ship when orientation is up

if V == 2
% Place the down pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R-1,C} = Boat_Mid_vert;
% Place the up pointing end of the boat at position (2,3)
Player_Board {R-2, C} = Boat_FrontBack_2;

end

%Set up ship when orientation is down
```

```
if V == 1

% Place the up pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions (2,4-6)
Player_Board{R+1,C} = Boat_Mid_vert;
% Place the down pointing end of the boat at position (2,3)
Player_Board {R+2, C} = Boat_FrontBack_1;

end

%Plot the board and show Cruiser

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)


%Clear the Command window

clc

%%


%Input coordinates of Destroyer and orientation

R = input ('What row do you want your Destroyer on? ');

C = input ('What column do you want your Destroyer on? ');

V = input ('What vertical orientation (Type 2 for up, 1 for down or 0 for none.) ');

H = input ('What horizontal orientation (Type 2 for left, 1 for right, or 0 for none.) ');

%Set up ship when orientation is right

if H == 1

% Place the left pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_3;
% Place the right pointing end of the boat at position (2,3)
Player_Board {R, C+1} = Boat_FrontBack_4;

end

%Set up ship when orientation is left

if H == 2

% Place the right pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_4;
% Place the left pointing end of the boat at position (2,3)
Player_Board {R, C-1} = Boat_FrontBack_3;

end

%Set up ship when orientation is up

if V == 2
% Place the down pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_1;
% Place the up pointing end of the boat at position (2,3)
Player_Board {R-1, C} = Boat_FrontBack_2;
```

```
end

%Set up ship when orientation is down

if V == 1

% Place the up pointing end of the boat at position (2,3)
Player_Board{R,C} = Boat_FrontBack_2;
% Place the down pointing end of the boat at position (2,3)
Player_Board {R+1, C} = Boat_FrontBack_1;

end

%Plot the board and show Destroyer

subplot(2,1,1); imshow([Opponent_Board{1,:};Opponent_Board{2,:};Opponent_Board{3,:};
Opponent_Board{4,:};Opponent_Board{5,:};Opponent_Board{6,:};
Opponent_Board{7,:};Opponent_Board{8,:};Opponent_Board{9,:};
Opponent_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)
subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

%Clear Command Window

clc



%%

%This entire section... Just Eww.

%Setup will determine the placement of the 5 ships that are part of the
%battleship game within the 10x10 gameboard.  The function will return a
%matrix where the 1st column of each row specifies the ship based on the
%list below, the second column specifies the direction (1 = vertical,
%2 = horizontal), and the third and fourth columns specify the starting
%coordinate of the ship.
%
%   Example-row 1 of Ships is: [2 1 4 5] --> specifies that the battlehsip
%   is placed vertically starting at position row 4, col 5 and would thus
%   occupy the following coordinates: (4,5), (5,5), (6,5), (7,5).
%                       ID#     Length
%   Aircraft Carrier    1       5
%   Battleship          2       4
%   Submarine           3       3
%   Cruiser             4       3
%   PT Boat             5       2
%
%   Usage: Ships = Setup() returns the setup information for the game board

% set up variables
Ships = zeros(5,4);
lengths = [5 4 3 3 2];
locations = zeros(5,10);

% begin placing ships, starting with the largest
number_of_ships = 1;
while number_of_ships <= 5
    % pick a coordinate
    coord = randi([1,10],[1,2]);
    % check to see if it is already used for another ship
    used = 0;
    for k = 1:5
        for m = 1:2:9
            if (coord(1,1) == locations(k,m) && coord(1,2) == locations(k,m+1))
                used = 1;
            end
        end
```

```
    end
    if used
        continue;
    end
    % check to see if the boat will fit using that point at one end
    use_check = [1 1 1 1]; % [left right up down]
    for k = 1:5
        for m = 1:2:9
            % check left
            if coord(1) == locations(k,m) && coord(2) >= locations(k,m+1) && coord(2) -
lengths(number_of_ships) <= locations(k,m+1) || coord(2) - lengths(number_of_ships) + 1 <= 0
                use_check(1) = 0;
            end
            % check right
            if coord(1) == locations(k,m) && coord(2) <= locations(k,m+1) && coord(2) +
lengths(number_of_ships) >= locations(k,m+1) || coord(2) + lengths(number_of_ships) - 1 > 10
                use_check(2) = 0;
            end
            % check up
            if coord(1) >= locations(k,m) && coord(1) - lengths(number_of_ships) <= locations(k,m) &&
coord(2) == locations(k,m+1) || coord(1) - lengths(number_of_ships) + 1 <= 0
                use_check(3) = 0;
            end
            % check down
            if coord(1) + lengths(number_of_ships) >= locations(k,m) && coord(1) <= locations(k,m) &&
coord(2) == locations(k,m+1) || coord(1) + lengths(number_of_ships) - 1 > 10
                use_check(4) = 0;
            end
        end
    end
    % boat does not fit
    if (use_check(1) == 0 && use_check(2) == 0 && use_check(3) == 0 && use_check(4) == 0)
        continue;
    % boat fits in at least one orientation, so pick an orientation and
    % place the boat
    else
        pick = randi([1,4],1);
        while (use_check(pick) ~= 1)
            pick = randi([1,4],1);
        end
        switch pick
            case 1  %left
                for k = 1:lengths(number_of_ships)
                    locations(number_of_ships,2*k-1) = coord(1);
                    locations(number_of_ships,2*k) = coord(2) - lengths(number_of_ships) + k;
                end
                Ships(number_of_ships,1) = number_of_ships;
                Ships(number_of_ships,2) = 2;
                Ships(number_of_ships,3) = coord(1);
                Ships(number_of_ships,4) = coord(2) - lengths(number_of_ships) + 1;
            case 2  %right
                for k = 1:lengths(number_of_ships)
                    locations(number_of_ships,2*k-1) = coord(1);
                    locations(number_of_ships,2*k) = coord(2) + k - 1;
                end
                Ships(number_of_ships,1) = number_of_ships;
                Ships(number_of_ships,2) = 2;
                Ships(number_of_ships,3) = coord(1);
                Ships(number_of_ships,4) = coord(2);
            case 3  %up
                for k = 1:lengths(number_of_ships)
                    locations(number_of_ships,2*k-1) = coord(1) - lengths(number_of_ships) + k;
                    locations(number_of_ships,2*k) = coord(2);
                end
                Ships(number_of_ships,1) = number_of_ships;
                Ships(number_of_ships,2) = 1;
                Ships(number_of_ships,3) = coord(1) - lengths(number_of_ships) + 1;
                Ships(number_of_ships,4) = coord(2);
            case 4  %down
                for k = 1:lengths(number_of_ships)
                    locations(number_of_ships,2*k-1) = coord(1) + k - 1;
                    locations(number_of_ships,2*k) = coord(2);
                end
                Ships(number_of_ships,1) = number_of_ships;
                Ships(number_of_ships,2) = 1;
                Ships(number_of_ships,3) = coord(1);
                Ships(number_of_ships,4) = coord(2);
        end
```

```
        number_of_ships = number_of_ships + 1;
    end

      COORD1 = coord(1,1); %Sets up row coordinates of various ships

      COORD2 = coord(1,2); %Sets up Column coordinate of various ships

      ORIENTATION = pick; %Sets up orientation of various ships

      %Combines all details of ship position into array
      %Because array is not previously defined, there must be a special
      %case to make it defined. In this case, it's when # of ships = 2

      if number_of_ships == 2

      %Defines an orientation and coordinate

      Array = [ORIENTATION,COORD1,COORD2];

      else

      %Defines more orientations and coordinates

      Array = [Array;ORIENTATION,COORD1,COORD2];

      end

end

ShipType = [1;2;3;4;5]; %Establishes different types of ships (Explained in intro of section)

CompShipPstn = [ShipType,Array]; %Finally spits out coordinates of all computer ships as an array

% Now Ships will be set up on the board w/ CompShipPstn as reference

%Column 1 - Ship Type
%Column 2 - Orientation (1 is left, 2 is right, 3 is up, 4 is down)
%Column 3 - Row Position
%Column 4 - Column Position

if CompShipPstn(1,2) == 1

% Place the right pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)-1} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)-2} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)-3} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)-4} = Boat_FrontBack_3;

end

if CompShipPstn(1,2) == 2

% Place the left-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)+1} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)+2} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)+3} = Boat_Mid_hor;
% Place the right-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)+4} = Boat_FrontBack_4;

end

if CompShipPstn(1,2) == 3

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(1,3)-1,CompShipPstn(1,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(1,3)-2,CompShipPstn(1,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(1,3)-3,CompShipPstn(1,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(1,3)-4,CompShipPstn(1,4)} = Boat_FrontBack_2;

end
```

```
if CompShipPstn(1,2) == 4

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(1,3),CompShipPstn(1,4)} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(1,3)+1,CompShipPstn(1,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(1,3)+2,CompShipPstn(1,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(1,3)+3,CompShipPstn(1,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(1,3)+4,CompShipPstn(1,4)} = Boat_FrontBack_1;

end

if CompShipPstn(2,2) == 1

% Place the right pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)-1} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)-2} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)-3} = Boat_FrontBack_3;

end

if CompShipPstn(2,2) == 2

% Place the left-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)+1} = Boat_Mid_hor;
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)+2} = Boat_Mid_hor;
% Place the right-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)+3} = Boat_FrontBack_4;

end

if CompShipPstn(2,2) == 3

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(2,3)-1,CompShipPstn(2,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(2,3)-2,CompShipPstn(2,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(2,3)-3,CompShipPstn(2,4)} = Boat_FrontBack_2;

end

if CompShipPstn(2,2) == 4

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(2,3),CompShipPstn(2,4)} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(2,3)+1,CompShipPstn(2,4)} = Boat_Mid_vert;
Opponent_Board{CompShipPstn(2,3)+2,CompShipPstn(2,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(2,3)+3,CompShipPstn(2,4)} = Boat_FrontBack_1;

end

if CompShipPstn(3,2) == 1

% Place the right pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)-1} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)-2} = Boat_FrontBack_3;

end

if CompShipPstn(3,2) == 2

% Place the left-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
```

```
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)+1} = Boat_Mid_hor;
% Place the right-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)+2} = Boat_FrontBack_4;

end

if CompShipPstn(3,2) == 3

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(3,3)-1,CompShipPstn(3,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(3,3)-2,CompShipPstn(3,4)} = Boat_FrontBack_2;

end

if CompShipPstn(3,2) == 4

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(3,3),CompShipPstn(3,4)} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(3,3)+1,CompShipPstn(3,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(3,3)+2,CompShipPstn(3,4)} = Boat_FrontBack_1;

end

if CompShipPstn(4,2) == 1

% Place the right pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)} = Boat_FrontBack_4;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)-1} = Boat_Mid_hor;
% Place the left pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)-2} = Boat_FrontBack_3;

end

if CompShipPstn(4,2) == 2

% Place the left-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)} = Boat_FrontBack_3;
% Place the middle sections of the boat at positions (2,4-6)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)+1} = Boat_Mid_hor;
% Place the right-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)+2} = Boat_FrontBack_4;

end

if CompShipPstn(4,2) == 3

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)} = Boat_FrontBack_1;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(4,3)-1,CompShipPstn(4,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(4,3)-2,CompShipPstn(4,4)} = Boat_FrontBack_2;

end

if CompShipPstn(4,2) == 4

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(4,3),CompShipPstn(4,4)} = Boat_FrontBack_2;
% Place the middle sections of the boat at positions
Opponent_Board{CompShipPstn(4,3)+1,CompShipPstn(4,4)} = Boat_Mid_vert;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(4,3)+2,CompShipPstn(4,4)} = Boat_FrontBack_1;

end

if CompShipPstn(5,2) == 1

% Place the right pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)} = Boat_FrontBack_4;
% Place the left pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)-1} = Boat_FrontBack_3;
```

```
end

if CompShipPstn(5,2) == 2

% Place the left-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)} = Boat_FrontBack_3;
% Place the right-pointing end of the boat at position (2,3)
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)+1} = Boat_FrontBack_4;

end

if CompShipPstn(5,2) == 3

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)} = Boat_FrontBack_1;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(5,3)-1,CompShipPstn(5,4)} = Boat_FrontBack_2;

end

if CompShipPstn(5,2) == 4

% Place the downward pointing end of the boat at position
Opponent_Board{CompShipPstn(5,3),CompShipPstn(5,4)} = Boat_FrontBack_2;
% Place the upward pointing end of the boat
Opponent_Board {CompShipPstn(5,3)+1,CompShipPstn(5,4)} = Boat_FrontBack_1;

end

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

%Now... Let's play. First, we must define hits and misses

H = 0; %Player Hits
M = 0; %Player Misses
CH = 0; %Computer Hits
CM = 0; %Computer Misses
CHMark = 0; %Variable to make MATLAB think about hits
%%

%Sets up while loop to start and end game
%Also, if for whatever reason the user goofs, that's no problem
%Just press Cntrl-Enter here, and the game will carry on like normal

%This while loop will stop if either the computer or the player wins

while H < 17 && CH < 17

    %Player Move

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Input Shot coordinates

    ROW = input ('Input Row # ');

    COLUMN = input ('Input Column # ');

    SHOT = [ROW,COLUMN];

    %If the coordinate has already been shot

    while X {SHOT (1,1), SHOT (1,2)} == Miss

        fprintf('\nThat square has already been bombed. Please try again\n\n')

         ROW = input ('Input Row # ');

    COLUMN = input ('Input Column # ');

    SHOT = [ROW,COLUMN];
```

```
    end
    %To prevent cheating by repeated bombings of a hit.
    while X {SHOT (1,1), SHOT (1,2)} == Hit
         fprintf('\nHey. No cheating.\n\n')
         ROW = input ('Input Row # ');
    COLUMN = input ('Input Column # ');
    SHOT = [ROW,COLUMN];
    end

    %Classifies the shot as either a hit or a miss
    if Opponent_Board{SHOT (1,1), SHOT (1,2)} == Open_Water
    X {SHOT (1,1), SHOT (1,2)} = Miss;
    M = M + 1;
    else
       X {SHOT (1,1), SHOT (1,2)} = Hit;
       H = H + 1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Computer move.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Makes sure that if player wins, the while loop is skipped
    if H < 17
       %Computer's first move
   if (CH || CM) == 0
       %Generates a random shot
    COMPSHOT = randi([1,10],[1,2]);
    CSStorage = COMPSHOT; %Another variable to make Matlab think
   end
    %Since MATLAB doesn't like dealing w/ cell images for whatever reason
    %We'll make it deal w/ 0s and 1s using Z, the cell filled w/ 0s

    %A majority of the computer's decision-making will go inside this loop
    %Since this helps the computer avoid bombing the same square
    while Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} ~= 0

        %If computer has no hit which would lead to tactical information,
        %Computer just shoots random points

        if CHMark == 0

            COMPSHOT = randi([1,10],[1,2]);

            %Just in case computer decides to go dumb and places CHMark = 0 at
            %this point, we know that if there's one square left, it has to
            %be next to another hit

            if CH == 16

                while Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} ~= 2
```

```
                    COMPSHOT = randi([1,10],[1,2]);

                end

                COMPSHOT = [COMPSHOT(1,1)+randi([-1,1]),COMPSHOT(1,2)+randi([-1,1])];

            end

        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %If computer successfully gets a hit which leads to tactical
    %information about a ship, it will hit surrounding squares as a first
    %Line of attach

         if CHMark == 1

             %if square above and square below are not available for
             %hitting, computer will hit adjacent squares in same row

            if (Z{COMPSHOT(1,1)+2,COMPSHOT(1,2)+1} ~= 0) && (Z{COMPSHOT(1,1),COMPSHOT(1,2)+1} ~= 0)

            COMPSHOT = [COMPSHOT(1,1),COMPSHOT(1,2)+randi([-1,1])];

            %Computer will prefer to test out columsn first

            else

            COMPSHOT = [COMPSHOT(1,1)+randi([-1,1]),COMPSHOT(1,2)];

            end

            %Since COMPSHOT has the chance to be 0 or 11 at this point
            %There will be safeguards to make sure this is not the case
            %When the final shot is made

            if COMPSHOT(1,1) == 0

            COMPSHOT(1,1) = CSStorage(1,1);

            elseif COMPSHOT(1,2) == 0

            COMPSHOT(1,2) = CSStorage(1,2);

            elseif COMPSHOT(1,1) == 11

                COMPSHOT(1,1) = CSStorage(1,1);

            elseif COMPSHOT(1,2) == 11

                COMPSHOT(1,2) = CSStorage(1,2);

            end

            %If all the adjacent squares have been bombed at this point,
            %for whatever reason, and no other hit has been made,
            %In order to protect the program, the computer will default
            %To making CHMark = 0

            if (Z{CSStorage(1,1)+2,CSStorage(1,2)+1} ~= 0) && (Z{CSStorage(1,1),CSStorage(1,2)+1} ~= 0)
&& (Z{CSStorage(1,1)+1,CSStorage(1,2)+2} ~= 0) && (Z{CSStorage(1,1)+1,CSStorage(1,2)} ~= 0)

                CHMark = 0;

            end

          end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%If a 2nd meaningful hit has been made, computer will adjust tactics to
%the situation appropriately

        if CHMark == 2
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This makes sure that computer does not handle any arguments within the
%danger-zone

            if COMPSHOT(1,1) == 1

            COMPSHOT(1,1) = CSStorage(1,1);

            elseif COMPSHOT(1,2) == 1

            COMPSHOT(1,2) = CSStorage(1,2);

            elseif COMPSHOT(1,1) == 10

            COMPSHOT(1,1) = CSStorage(1,1);

            elseif COMPSHOT(1,2) == 10

            COMPSHOT(1,2) = CSStorage(1,2);

            end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%If there is a hit on the square below, but no hit on the square above

         if (Z{COMPSHOT(1,1)+2,COMPSHOT(1,2)+1} == 2) && (Z{COMPSHOT(1,1),COMPSHOT(1,2)+1} == 0)

              COMPSHOT = [COMPSHOT(1,1)-1, COMPSHOT(1,2)];

%If there is a hit on the square above, but no hit on the square below

         elseif (Z{COMPSHOT(1,1),COMPSHOT(1,2)+1} == 2) && (Z{COMPSHOT(1,1)+2,COMPSHOT(1,2)+1} == 0)

           COMPSHOT = [COMPSHOT(1,1)+1, COMPSHOT(1,2)];

%If there is a hit on the square right, but no hit on the square left

         elseif (Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)+2} == 2) && (Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)} == 0)

             COMPSHOT = [COMPSHOT(1,1), COMPSHOT(1,2)-1];

%If there is a hit on the square left

         elseif (Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)} == 2) && (Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)+2} == 0)

             COMPSHOT = [COMPSHOT(1,1),COMPSHOT(1,2)+1];

         end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %If, for whatever reason, the square the computer chooses is
        %occupied, in order to protect the program, the computer will
        %Resort to bombing points at random


            if Z{COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} ~= 0

                CHMark = 0;

            end

       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        end

 %Note, when computer misses, it will make COMPSHOT == CSStorage
 %Which, when going back to this code, provides successful
 %continuation of smart battleship play (AKA, carpet bomb
 %opposite direction when a miss is encountered at this stage)
 %This will become apparent later on in the code

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %If, for whatever reason, CHMark starts to = 3, in order to protect
 %the program, CHMark will = 0 at this stage
```

```
if CHMark == 3

    CHMark = 0;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   end

%CCS Storage is "Computer Shot Storage"
%If a hit is made, this will store the coordinate of hit for later ref.

 if CHMark == 0

 CSStorage = COMPSHOT; %Saves coordinate for CHMarks

 end

   %If computer shoots open water, that's a miss and computer adjusts
   %next compshot to Computer shot storage

  if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Open_Water

       Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Miss;

       CM = CM + 1;

       Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 1;

       COMPSHOT = CSStorage;

  end

  %If computer shoots a part of the boat
  %The computer will show that part hit
  %If CHMark = 2, There is no need to keep adding to CHMark
  %Since there are appropriate safeguards to get computer out of that
  %stage of attack

   if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_FrontBack_1

       Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_FrontBack_1_hit;

       CH = CH + 1;

       Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;

       if CHMark <= 1

       CHMark = CHMark + 1;

       end

   end

    if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_FrontBack_2

       Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_FrontBack_2_hit;

       CH = CH + 1;

       Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;

       if CHMark <= 1

       CHMark = CHMark + 1;

       end

    end

    if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_FrontBack_3

       Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_FrontBack_3_hit;

       CH = CH + 1;
```

```
        Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;
        if CHMark <= 1
        CHMark = CHMark + 1;
        end
    end
    if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_FrontBack_4
        Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_FrontBack_4_hit;
        CH = CH + 1;
        Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;
        if CHMark <=1
        CHMark = CHMark + 1;
        end
    end
    if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_Mid_hor
        Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_Mid_hor_hit;
        CH = CH + 1;
        Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;
        if CHMark <= 1
        CHMark = CHMark + 1;
        end
    end
    if Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} == Boat_Mid_vert
        Player_Board {COMPSHOT(1,1),COMPSHOT(1,2)} = Boat_Mid_vert_hit;
        CH = CH + 1;
        Z {COMPSHOT(1,1)+1,COMPSHOT(1,2)+1} = 2;
        if CHMark <= 1
        CHMark = CHMark + 1;
        end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Show the aftermath of attacks from both computer and player

subplot(2,1,1); imshow([X{1,:};X{2,:};X{3,:};
X{4,:};X{5,:};X{6,:};
X{7,:};X{8,:};X{9,:};
X{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)
```

```
ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

clc

    end

end

%If the player won, shows the aftermath and prints out win message
%w/ # of shots

if H == 17

subplot(2,1,1); imshow([X{1,:};X{2,:};X{3,:};
X{4,:};X{5,:};X{6,:};
X{7,:};X{8,:};X{9,:};
X{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

clc

fprintf ('\nCongratulations, player! You won in %i shots!\n',(H+M))

end

%If the computer won, shows the aftermath and prints out loss message
%w/ # of shots

if CH == 17

    subplot(2,1,1); imshow([X{1,:};X{2,:};X{3,:};
X{4,:};X{5,:};X{6,:};
X{7,:};X{8,:};X{9,:};
X{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

subplot(2,1,2); imshow([Player_Board{1,:};Player_Board{2,:};Player_Board{3,:};
Player_Board{4,:};Player_Board{5,:};Player_Board{6,:};Player_Board{7,:};
Player_Board{8,:};Player_Board{9,:};Player_Board{10,:}]);

xlabel('1 2 3 4 5 6 7 8 9 10','fontsize', 28)

ylabel('10 9 8 7 6 5 4 3 2 1','fontsize', 28)

clc

    fprintf('\nAw, no! You lost in %i shots! Better luck next time.\n', (CH+CM))

end
```