Project 2

<Battleship - Abridged and Virtual V2>

CSC5-48101

Name: Cody Steimle

Date: 12/12/16

Introduction:

Title: Battleship - Abridged and Virtual

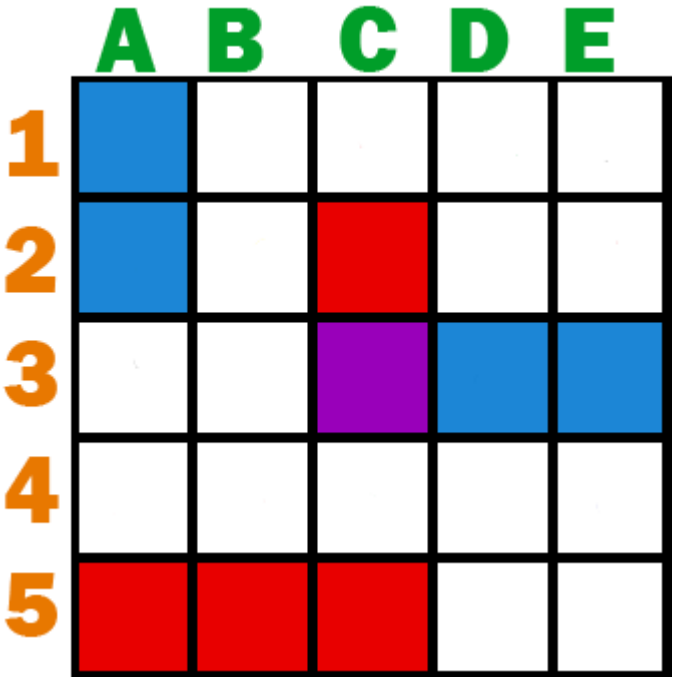
Just like the popular board game, this Battleship program emulates the original board game with a few differences. The three differences are: instead of the board being a 10x10, it is cut in half to a 5x5 virtual board; the ships are affected as well, instead of the 5+ ship configuration in the original, I've simplified it to only 2 ships to avoid too much confusion from a lack of visuals (a 1x2 ship and a 1x3 ship); lastly, the player may choose where and how to place the first ship, but the second one is randomized to avoid accidental overlapping.

The guesses in the game come in two parts, the column and row guess. They are either entered one at a time (A -> return -> 1), or together (A1). However the letters are case sensitive, so I've added input validation to insure that the user enters a proper value.

Before the game begins a coin toss is done to find out who will be taking the first turn. After that is decided the game will alternate between players as they take their turns. The turns are kept track of and as the last ship is sunk the game will end by breaking out of the loop and finishing out the program.

Version 2 additions: Added a board for the player to view to keep track of the opponent's ship easier using a 2D array. Created more functions to clean up main a bit, one being savGame to output results to a file. Fixed a bug where after the comp won it wouldn't break out of the loop.

*Visual example



Development Summary:

Project size: about 650 Lines
Number of variables: about 40

Version 2 update: The biggest improvement to the program is the 2D array that displays the player's view of the game. I was really pleased to get it working while being spaced out and organized, as well as the column letters and row numbers not being included in the actual array but still visible. Also finally got the results to save properly to a file.

This project started off very smooth and most of the ground work didn't take much longer than a day. However after having a working program with one ship, I had to completely rework the code to make the second ship. It took deep thinking to figure out how to make the two ships not intersect each other, while still remaining inside the 5x5 grid. The solution came to me while adjusting the random outputs and testing out nesting. From this project I learned more about how the data types interact with each other (ex. char and int), and new ways to make logical operators work.

Along with the data types, I also used a small function I created to calculate the computer turn without having repost code (comAtk seen underneath main). The most used construct in my program is loops. They might consist of 80% of the code, and I have used do-while loops like in line 93-112, as well as if-else (if-if else) loops given in lines 123-139 for example.

There is randomized computer placement and guesses so no 2 games will be alike. Also the Player may set a name that will be used throughout the game.

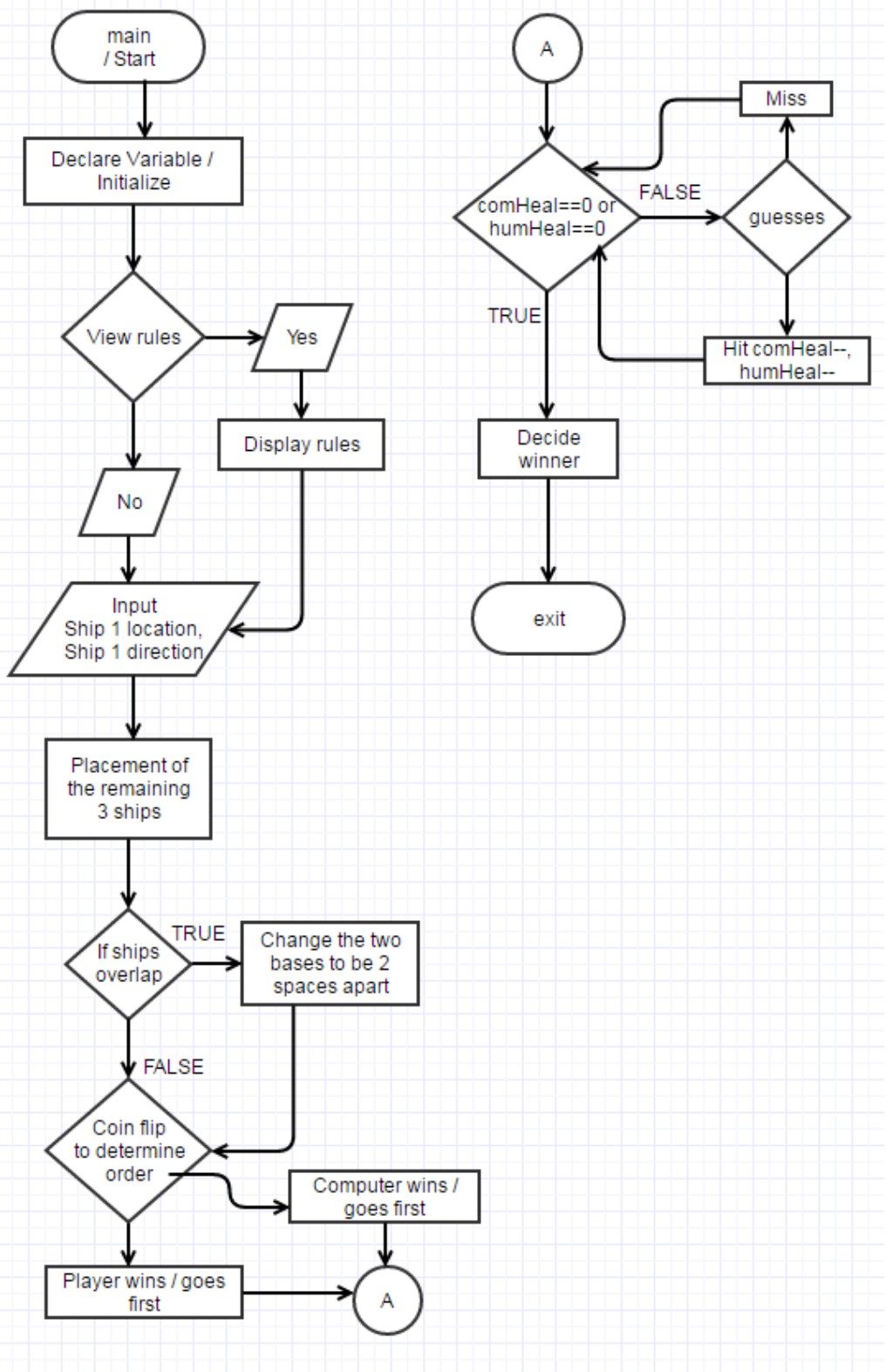
Chapter Concepts:

Textbook	Chap.Sec	Concept	Code Location
Savitch 9th	1.3	#include	Lines 9-13
Savitch 9th	1.3	namespace	Line 14
Savitch 9th	1.3	return	Lines 508,598
Savitch 9th	2.1	variable	Lines 36-65
Savitch 9th	2.1	variable assignments	Lines 42-44
Savitch 9th	2.1	identifiers	Lines 36-65
Savitch 9th	2.2	output/cout	Line 69
Savitch 9th	2.2	input/cin	Line 77
Savitch 9th	2.2	escape sequence	Line 70 (endl)
Savitch 9th	2.3	int data types	Lines 44, 350
Savitch 9th	2.3	char data types	Lines 49, 87
Savitch 9th	2.3	bool data types	Lines 46, 489

Savitch 9th	2.3	strings	Lines 36, 78
Savitch 9th	2.3	math expressions	Line 120
Savitch 9th	2.4	if-else branching	Lines 109-117
Savitch 9th	2.4	operators	Lines 88, 92
Savitch 9th	2.4	do-while loop	Line 154-157
Savitch 9th	2.5	comments	Lines 35-65
Savitch 9th	2.5	indenting	Line 86
Savitch 9th	3.1	boolean expressions	Lines 88, 92
Savitch 9th	3.2	nested statements	Lines 109-126
Savitch 9th	3.2	multi if-else statements	Lines 109-126
Savitch 9th	3.2	switch statement	Lines 99-108
Savitch 9th	3.2	blocks	Lines 176-178 (inside if-statement)
Savitch 9th	3.3	increment / decrement	Lines 350
Savitch 9th	3.3	for loops	Lines 664-668
Savitch 9th	3.3	break in a loop	Line 438
Savitch 9th	3.4	nested loops	Line 371-385
Savitch 9th	4.2	predefined functions	Lines 129-130 (rand)
Savitch 9th	4.2	random number gen.	Line 129
Savitch 9th	4.2	type casting	Line 256 (testing char as an int)
Savitch 9th	4.3	function call	Line 73
Savitch 9th	4.3	function return	Line 598
Savitch 9th	4.5	global constants	Lines 19-20 (for 2D array)
Savitch 9th	4.6	function prototypes	Lines 23-28
Savitch 9th	5.1	void functions	Line 23
Savitch 9th	5.2	reference parameters	Lines 25
Savitch 9th	6.1	file input / output	Lines 610-638
Gaddis 8th	4.11	validating user input	Line 572
Gaddis 8th	6.5	passing by value	Line 352
Gaddis 8th	6.8	returning a value	Line 522
Gaddis 8th	6.9	returning a boolean	Line 598
Gaddis 8th	6.12	default arguments	Line 42
Gaddis 8th	6.13	passing by reference	Line 517
Gaddis 8th	7.1	arrays	Lines 645-648
Gaddis 8th	7.2	access array	Line 667
Gaddis 8th	7.3	in-bounds of arrays	Line 665 (starting at 0 not 1)
Gaddis 8th	7.5	processing arrays	Lines 665-667
Gaddis 8th	7.7	passing arrays with functions	Line 658
Gaddis 8th	7.8	2 dimensional arrays	Line 47
Gaddis 8th	7.12	vectors	N/A
Gaddis 8th	8.1	linear / binary search	N/A
Gaddis 8th	8.3	bubble / selection sort	N/A

Gaddis 8th	8.5	sorting / searching vectors	N/A
Gaddis 8th	9.1	printing address of variable	N/A
Gaddis 8th	9.2	pointer variables	N/A

Flowchart:


main
/ Start
Declare Variable /
Initialize
View rules
Yes
Display rules
No
Input
Ship 1 location,
Ship 1 direction
Placement of
the remaining
3 ships
If ships
overlap
TRUE
Change the two
bases to be 2
spaces apart
FALSE
Coin flip
to determine
order
Computer wins /
goes first
Player wins / goes
first
A
A
Miss
comHeal==0 or
humHeal==0
FALSE
guesses
TRUE
Hit comHeal--,
humHeal--
Decide
winner
exit

Pseudo Code:

Initialize

Ask if the user would like to view the rules

If the user wishes to show rules

    Show the rules

Else continue to the placement of the four ships

Fill the game board

Prompt the user to set the 1x2 piece.

Randomize the placement of the second piece

    If the piece overlaps

        Shift the base two spaces apart

    Else begin setting the Comp pieces

Randomize the 1x2 Comp piece

Randomize the 1x3 Comp piece until no overlap

Coin flip calculation to see if Comp or Player goes first

If Comp wins

    Comp guess first

Else Player Wins

    Skip to Player's turn

Both continue to guess until all pieces of the ships on a single side have been guessed.

Calculate and display the winner.

Prompt the user to save results to file

Exit Program.

Variables:

Data Type	Name	Location	Description
const int	ROWS=5	global	Number of rows for game board
const int	COLS=5	global	Number of columns for game board
string	Line	main	Place holder line
char	compCol	main	Computer's column guess
char	compRow	main	Computer's row guess

char	chosDir	main	The ship direction of a player's choice
int	shipDir	main	Ship direction, which will be randomized
char	human	main	Basic character input from the player
char	humCol	main	Player's column guess
char	humRow	main	Player's row guess
char	comHeal='5'	main	Total health for the computer
char	humHeal='5'	main	Total health for the human
int	turn=0	main	The turn counter, starts at zero
string	name	main	The name of the player
bool	winner	main	Winning player (0 - Computer, 1 - Player)
char	board[ROWS][COLS]	main	The board array
char	hsmlC1, hsmlR1	main	Column+row for human small ship: part 1
char	hsmlC2, hsmlR2	main	Column+row for human small ship: part 2
char	hsmlDr	main	Horizontal or vertical for human small ship
char	csmlC1, csmlR1	main	Column+row for computer small ship: part 1
char	csmlC2, csmlR2	main	Column+row for computer small ship: part 2
char	csmlDr	main	Horizontal or vertical for computer small ship
char	hmedC1, hmedR1	main	Column+row for human medium ship: part 1
char	hmedC2, hmedR2	main	Column+row for human medium ship: part 2
char	hmedC3, hmedR3	main	Column+row for human medium ship: part 3
char	hmedDr	main	Horizontal/ vertical for human medium ship
char	cmedC1, cmedR1	main	Column+row for computer medium ship: p.1
char	cmedC2, cmedR2	main	Column+row for computer medium ship: p.2
char	cmedC3, cmedR3	main	Column+row for comp medium ship: p.3
char	cmedDr	main	Horizontal/vertical for comp medium ship
int	count	prntAry	Tracks the array print
char	human	disRule	Basic character input from the player
int	Coin	coinFlp	The coin, the value will equal heads/tails (1/2).
bool	isWon	coinFlp	Checks if player won the coin flip
char	human	coinFlp	Basic character input from the player
ifstream	In	savGame	Input file
ofstream	Out	savGame	Output file
string	results	savGame	Holder to print the save file
char	human	savGame	Basic character input from the player

Functions:

Type Name	Purpose	Inputs	Outputs
char comAtk	Randomizes computer guess	compCol, compRow	compCol, compRow
void disRule	Displays the rules	human	Game rules
void fillAry	Fill in the game board	None	None

void prntAry	Print the board	None	board[][] array
bool coinFlp	Coin flip for turn order	human	Winner of flip
void savGame	Prompts to output to file	human	Save game file

Reference:

1. Textbook (Savitch 9th Edition).
2. Textbook (Gaddis 8th Edition).
3. Mark Lehr GitHub repository.
4. Official Battleship board game.

Program (main):

```
//Set random number seed
   srand(static_cast<unsigned int>(time(0)));

   //Declaration of Variables
   string line;          //Place holder line
   char compCol, compRow;  //Computer Attack
   int shipDir;          //Ship direction for random
   char chosDir;          //Ship direction
   char human;           //Player inputs
   char humCol, humRow;    //Player's Attack
   char comHeal='5';      //Total computer health
   char humHeal='5';      //Total player health
   int turn=0;           //Tracks turn count
   string name;          //Player's Name
   bool winner;          //Winning player (0 - Computer, 1 - Player)
   char board[ROWS][COLS]; //The board
   //Small ship Human
   char hsmlC1, hsmlR1;   //1x2 Base
   char hsmlC2, hsmlR2;   //1x2 Tail
   char hsmlDr;          //Direction
   //Small ship Comp
   char csmlC1, csmlR1;   //1x2 Base
   char csmlC2, csmlR2;   //1x2 Tail
   char csmlDr;          //Direction
   //Medium ship Human
   char hmedC1, hmedR1;   //1x3 Base
   char hmedC2, hmedR2;   //1x3 Tail
   char hmedC3, hmedR3;   //1x3 Tail
   char hmedDr;          //Direction
   //Medium ship Comp
   char cmedC1, cmedR1;   //1x3 Base
   char cmedC2, cmedR2;   //1x3 Tail
```

```
char cmedC3, cmedR3;   //1x3 Tail
char cmedDr;           //Direction


//Initialize
cout<<"Welcome to the program that emulates a abridged version of the "
   "classic board game, Battleship."<<endl;

//Rules
disRule();

//Game Setup
cout<<"What is your desired name? (no spaces)"<<endl;
cin>>name; //Input Name
cout<<"Alright then "<<name<<". Get ready for a game of Battleship!"<<endl;

//Set an empty board
fillAry(board);

//Player's Small Ship
cout<<"Okay, now choose the placement of the smaller ship. (1x2)"<<endl;
do{    //Validate Input
   cout<<"Choose a column (A-E, case sensitive):";
   cin>>hsmlC1;
}while(hsmlC1<65||hsmlC1>69);
do{    //Validate Input
   cout<<"Choose a row (1-5):";
   cin>>hsmlR1;
}while(hsmlR1<49||hsmlR1>53);

//Direction of Player's Small Ship
do{  //Validate Input
   cout<<"Now choose either Vertical (V) or Horizontal (H):";
   cin>>chosDir;
}while(chosDir!='h'&&chosDir!='H'&&chosDir!='v'&&chosDir!='V');
switch(chosDir){
   case 'h':
   case 'H': {
      cout<<"You picked Horizontal."<<endl;
      hsmlDr='H';};break;
   case 'v':
   case 'V': {
      cout<<"You picked Vertical."<<endl;
      hsmlDr='V';}
```

```
}
if(hsmlDr=='V'){    //Vertical
   hsmlC2=hsmlC1;
   if(hsmlR1+1>53){  //Keep the ship inside the board
      hsmlC2=hsmlC1-1;
   }
   else{
      hsmlR2=hsmlR1+1;
   }
}
else if(hsmlDr=='H'){  //Horizontal
   if(hsmlC1+1>69){
      hsmlC2=hsmlC1-1;
   }
   else{
      hsmlC2=hsmlC1+1;
   }
   hsmlR2=hsmlR1;
}

//Computer's Small Ship
csmlC1=rand()%5+65;      //Calls random function, then modifies to A-E
csmlR1=rand()%5+49;      //Calls random function, then modifies to 1-5
//Direction of Computer's Small Ship
shipDir=rand()%10+1;
if(shipDir<=5){    //Vertical
   csmlDr='V';
   csmlC2=csmlC1;
   if(csmlR1+1>53){
      csmlR2=csmlR1-1;
   }
   else{
      csmlR2=csmlR1+1;
   }
}
else if(shipDir>5){  //Horizontal
   csmlDr='H';
   if(csmlC1+1>69){  //Keep the ship inside the board
      csmlC2=csmlC1-1;
   }
   else{
      csmlC2=csmlC1+1;
   }
   csmlR2=csmlR1;
```

```
}
//Computer's Medium Ship
do{
   cmedC1=rand()%5+65;      //Calls random function, then modifies to A-E
   cmedR1=rand()%5+49;      //Calls random function, then modifies to 1-5
}while(cmedC1==csmlC1||cmedR1==csmlR1);
//Direction of Computer's Medium Ship
shipDir=rand()%10+1;
if(shipDir<=5){  //Vertical
   cmedDr='V';
   if(cmedDr==csmlDr){  //If the directions are the same
      if(csmlC1==cmedC1){
         if(cmedC1+1>69){  //Keep the ship inside the board
            cmedC1=cmedC1-1;
         }
         else{
            cmedC1=cmedC1+1;
         }
      }
      if(cmedR1+2>53){  //Keep the ship inside the board
         cmedR2=cmedR1-1;
         cmedR3=cmedR2-1;
      }
      else{
         cmedR2=cmedR1+1;
         cmedR3=cmedR2+1;
      }
      cmedC2=cmedC1;
      cmedC3=cmedC1;
   }
   else{
      if(csmlC1+2>69){  //Keep the ship inside the board
         cmedC1=csmlC1-2;
      }
      else{
         cmedC1=csmlC1+2;
      }
      if(cmedR1+2>53){  //Keep the ship inside the board
         cmedR2=cmedR1-1;
         cmedR3=cmedR2-1;
      }
      else{
         cmedR2=cmedR1+1;
         cmedR3=cmedR2+1;
```

```
      }
      cmedC2=cmedC1;
      cmedC3=cmedC1;
    }
  }
  else if(shipDir>5){  //Horizontal
    cmedDr='H';
    if(cmedDr==csmlDr){  //If the directions are the same
      if(csmlR1==cmedR1){
        if(cmedR1+1>53){  //Keep the ship inside the board
          cmedR1=cmedR1-1;
        }
        else{
          cmedR1=cmedR1+1;
        }
      }
      if(cmedC1+2>69){  //Keep the ship inside the board
        cmedC2=cmedC1-1;
        cmedC3=cmedC2-1;
      }
      else{
        cmedC2=cmedC1+1;
        cmedC3=cmedC2+1;
      }
      cmedR2=cmedR1;
      cmedR3=cmedR1;
    }
    else{
      if(csmlR1+2>53){  //Keep the ship inside the board
        cmedR1=csmlR1-2;
      }
      else{
        cmedR1=csmlR1+2;
      }
      if(cmedC1+2>69){  //Keep the ship inside the board
        cmedC2=cmedC1-1;
        cmedC3=cmedC2-1;
      }
      else{
        cmedC2=cmedC1+1;
        cmedC3=cmedC2+1;
      }
      cmedR2=cmedR1;
      cmedR3=cmedR1;
```

```
   }
}

//Player's Medium Ship
cout<<"Okay, now the placement of the second ship will be random to prevent"
      " overlapping of pieces."<<endl;
 do{
   hmedC1=rand()%5+65;      //Calls random function, then modifies to A-E
   hmedR1=rand()%5+49;      //Calls random function, then modifies to 1-5
}while(hmedC1==hsmlC1||hmedR1==hsmlR1);
//Direction of Computer's Medium Ship
shipDir=rand()%10+1;
if(shipDir<=5){  //Vertical
   hmedDr='V';
   if(hmedDr==hsmlDr){  //If the directions are the same
      if(hsmlC1==hmedC1){
         if(hmedC1+1>69){  //Keep the ship inside the board
            hmedC1=hmedC1-1;
         }
         else{
            hmedC1=hmedC1+1;
         }
      }
      if(hmedR1+2>53){
         hmedR2=hmedR1-1;
         hmedR3=hmedR2-1;
      }
      else{
         hmedR2=hmedR1+1;
         hmedR3=hmedR2+1;
      }
      hmedC2=hmedC1;
      hmedC3=hmedC1;
   }
   else{
      if(hsmlC1+2>69){  //Keep the ship inside the board
         hmedC1=hsmlC1-2;
      }
      else{
         hmedC1=hsmlC1+2;
      }
      if(hmedR1+2>53){  //Keep the ship inside the board
         hmedR2=hmedR1-1;
         hmedR3=hmedR2-1;
```

```
      }
      else{
         hmedR2=hmedR1+1;
         hmedR3=hmedR2+1;
      }
      hmedC2=hmedC1;
      hmedC3=hmedC1;
   }
}
else if(shipDir>5){  //Horizontal
   hmedDr='H';
   if(hmedDr==hsmlDr){  //If the directions are the same
      if(hsmlR1==hmedR1){
         if(hmedR1+1>53){
            hmedR1=hmedR1-1;
         }
         else{
            hmedR1=hmedR1+1;
         }
      }
      if(hmedC1+2>69){  //Keep the ship inside the board
         hmedC2=hmedC1-1;
         hmedC3=hmedC2-1;
      }
      else{
         hmedC2=hmedC1+1;
         hmedC3=hmedC2+1;
      }
      hmedR2=hmedR1;
      hmedR3=hmedR1;
   }
   else{
      if(hsmlR1+2>53){  //Keep the ship inside the board
         hmedR1=hsmlR1-2;
      }
      else{
         hmedR1=hsmlR1+2;
      }
      if(hmedC1+2>69){  //Keep the ship inside the board
         hmedC2=hmedC1-1;
         hmedC3=hmedC2-1;
      }
      else{
         hmedC2=hmedC1+1;
```

```
        hmedC3=hmedC2+1;
      }
      hmedR2=hmedR1;
      hmedR3=hmedR1;
   }
}

//Display ship locations
cout<<"------------------------------------------------------------"<<endl;
cout<<name<<", your ship locations are:"<<endl;
cout<<"Small: "<<hsmlC1<<hsmlR1<<" "<<hsmlC2<<hsmlR2<<endl;
cout<<"Medium: "<<hmedC1<<hmedR1<<" "<<hmedC2<<hmedR2<<" "<<hmedC3<<hmedR3
   <<endl;
cout<<"------------------------------------------------------------"<<endl;

//Game Start!
if(coinFlp()==1){
   cout<<name<<" will go first."<<endl;
}
else{
   cout<<"Computer will go first."<<endl;
   cout<<endl;
   turn++;
   cout<<"                Turn #"<<turn<<endl;
   comAtk(compCol, compRow);
   if(compCol==hsmlC1&&compRow==hsmlR1){
      cout<<"HIT!"<<endl;
      humHeal--;
      hsmlC1='0';
      hsmlR1='0';
   }
   else if(compCol==hsmlC2&&compRow==hsmlR2){
      cout<<"HIT!"<<endl;
      humHeal--;
      hsmlC2='0';
      hsmlR2='0';
   }
   else{
      cout<<"Miss..."<<endl;
   }
   cout<<name<<"'s remaining health: "<<humHeal<<endl; //Displays User HP
}

do{
```

```
//Player's Turn
cout<<endl;
turn++;
cout<<"                Turn #"<<turn<<endl;
cout<<"------------------"<<name<<"'s Turn------------------"<<endl;
cout<<"Make your guess."<<endl;
do{    //Validate Input
   cout<<"Choose a column (A-E, case sensitive):"<<endl;
   cin>>humCol;
}while(humCol<65||humCol>69); //65 is char 'A', 69 is char 'E'
do{    //Validate Input
   cout<<"Choose a row (1-5):"<<endl;
   cin>>humRow;
}while(humRow<49||humRow>53);  //49 is char '1', 53 is char '5'

cout<<name<<" attacks "<<humCol<<"-"<<humRow<<"!"<<endl;

if(humCol==csmlC1&&humRow==csmlR1){
   cout<<"HIT!"<<endl;
   humRow=humRow-49; //Converts to 0-4
   humCol=humCol-65; //Converts to 0-4
   board[humRow][humCol]='X'; //Marker
   comHeal--;
   csmlC1='0';
   csmlR1='0';
}
else if(humCol==csmlC2&&humRow==csmlR2){
   cout<<"HIT!"<<endl;
   humRow=humRow-49; //Converts to 0-4
   humCol=humCol-65; //Converts to 0-4
   board[humRow][humCol]='X'; //Marker
   comHeal--;
   csmlC2='0';
   csmlR2='0';
}
else if(humCol==cmedC1&&humRow==cmedR1){
   cout<<"HIT!"<<endl;
   humRow=humRow-49; //Converts to 0-4
   humCol=humCol-65; //Converts to 0-4
   board[humRow][humCol]='X'; //Marker
   comHeal--;
   cmedC1='0';
   cmedR1='0';
}
```

```
else if(humCol==cmedC2&&humRow==cmedR2){
   cout<<"HIT!"<<endl;
   humRow=humRow-49; //Converts to 0-4
   humCol=humCol-65; //Converts to 0-4
   board[humRow][humCol]='X'; //Marker
   comHeal--;
   cmedC2='0';
   cmedR2='0';
}
else if(humCol==cmedC3&&humRow==cmedR3){
   cout<<"HIT!"<<endl;
   humRow=humRow-49; //Converts to 0-4
   humCol=humCol-65; //Converts to 0-4
   board[humRow][humCol]='X'; //Marker
   comHeal--;
   cmedC3='0';
   cmedR3='0';
}
else{
   cout<<"Miss..."<<endl;
}
prntAry(board); //Print the board
if(comHeal=='0')break;  //Breaks out of the code when the computer dies

//Computer Turn
cout<<endl;
turn++;
cout<<"                Turn #"<<turn<<endl;
comAtk(compCol, compRow);
if(compCol==hsmlC1&&compRow==hsmlR1){
   cout<<"HIT!"<<endl;
   humHeal--;
   hsmlC1='0';
   hsmlR1='0';
}
else if(compCol==hsmlC2&&compRow==hsmlR2){
   cout<<"HIT!"<<endl;
   humHeal--;
   hsmlC2='0';
   hsmlR2='0';
}
else if(compCol==hmedC1&&compRow==hmedR1){
   cout<<"HIT!"<<endl;
   humHeal--;
```

```
      hmedC1='0';
      hmedR1='0';
   }
   else if(compCol==hmedC2&&compRow==hmedR2){
      cout<<"HIT!"<<endl;
      humHeal--;
      hmedC2='0';
      hmedR2='0';
   }
   else if(compCol==hmedC3&&compRow==hmedR3){
      cout<<"HIT!"<<endl;
      humHeal--;
      hmedC3='0';
      hmedR3='0';
   }
   else{
   cout<<"Miss..."<<endl;
   }
   cout<<name<<"'s remaining health: "<<humHeal<<endl; //Displays User HP
}while(humHeal!='0'); //Breaks out of the code when the users dies

//Post Game
cout<<endl;
if((comHeal=='0')&&(humHeal!='0')){
   cout<<"*****"<<name<<" Won!*****"<<endl;
   winner=1;
}
else{
   cout<<"*****You Lose.*****"<<endl;
   winner=0;
}
cout<<endl;

//Store Winner
if(winner==0){
   line='Comp';
}
if(winner==1){
   line=name;
}

//Results
savGame(turn, line);
cout<<"************************************************************"<<endl;
```

```
    cout<<"*                Thanks for playing!                *"<<endl;
    cout<<"***********************************************************"<<endl;

    //Exit Program
    return 0;
}
```