Project-2

Title

Battleship Game

Course

CSC 17A

Section

48290

Due Date

12/11/2022

Author

Khadiza Akter

Contents

1. Introduction:

Battleship is a two-player strategy guessing game commonly referred to as Battleships or Sea Battle. Each player's fleet of warships is marked on regulated grids (on paper or on a board) on which the game is played. The enemy player cannot see where the ships are located. The goal of the game is to wipe out the rival player's fleet by taking turns calling "shots" at one other's ships.

The game is played on two grids, two for each player. The grids are typically square – usually 10×10 – and the individual squares in the grid are identified by letter and number. If a player successfully hit the opponent ship, then it is marked as different letter

Each participant discreetly sets their ships on their grid before the game starts. Each ship is positioned on the grid in a series of parallel locations, either horizontally or vertically. The number of positions for each ship is determined by the type of ship. The vessels cannot converge (i.e., only one ship can occupy any given position in the grid). The types and numbers of ships allowed are the same for each player. The ships should be hidden from players sight and it's not allowed to see each other's pieces. The game is a discovery game which player need to discover their opponents ship positions. The play who is able to destroy all the ships first then he/she is the winner of the game [1].

In this game, we consider 4 type of ships that are shown in Table1:

Table 1. Type of ships

Class of ships	Size
Aircraft	5
Battleship	4
Destroyer	3
Corvette	2

2. Development Summary

The project was developed using six different versions.

GameOfBattleShip_V1:

In version 1, the project was started and filled-up the 2-D vector for player and computer, and developed the function to display the game rules.

GameOfBattleShip _V2:

In version 2, the player was able to setup the ships in his/her matrix position.

GameOfBattleShip_V3:

In version 3, the computer was able to setup the ships randomly.

GameOfBattleShip_V4:

In version 4, the program was able to display the play zone for both computer and player. The play can start using user input, and computer can select an attack position randomly as well.

GameOfBattleShip_V5:

In version 5, check the player and computer wheather their attack was successfully or not. Update both player matrix and computer matrix for indicating miss hit as ‘o’ and sucessful hit as ‘@’. Also, in this version updated the ship status for computer and player and displayed status information at the top of the screen. Finally, this version determined the winner of the game.

GameOfBattleShip_V6:

In this version, the program was able to save a game, and opened a save game successfully.

GameOfBattleShip_V7_Final:

The program was able to show the last game summary and last winner of the game. Also, this version was finalized the menu of the game, updating the comments, formats, lines requirement (breaking the long lines), testing the game.
Lines of code: 1807 (Including Spaces and Comments).

Lines of Code: 1537

Number of variables: 84 (Approximate)

Number of methods: 27

I worked on the project for around four weeks and spent around 160 hours. I applied several concepts from chapters 13 to 15 to complete the project and learned how to use these concepts in a software project. Apart from these chapters, I had to add concepts such as random variables, method overloading, and current date-time functions.

3. Description

3.1 Game Rules

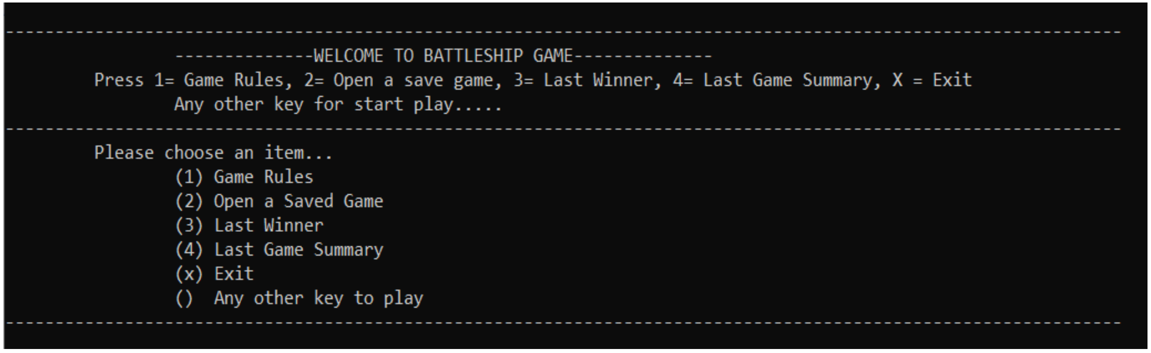
The game rules are summarised as follows:

1. Total four battleships for each player, the winner is who destroy other battleships first
2. The battlefield is 10x10 grid where you place all four ships
3. You can place your ships position using coordinate values (e.g., A0, B1) where 'A' or 'a' is the row and 1-10 is the column number
4. Also, you can place the ship orientation, i. e, horizontal or vertical. For horizontal orientation, type 'h' or 'H', and type 'v' or 'V' for vertical option.

5. You have total four battle ships: Aircraft Carrier-> 5 Battleship-> 4, Destroyer-> 3 and Corvette-> 2 units long.
6. You cannot place two ship at any same coordinate location.
7. After placing your ship position; you are ready to play. To attack the opponent, enter a position value such as A1 or a1, b9, j5 (without spacing) and so on.
8. If your attack is successful then it is denoted by '@', and you will continue your turn
9. If your attack is missed then it is denoted by 'o', and your turn will be end and computer will attack your ships.

3.2 How to Play the Game (Input/Output)

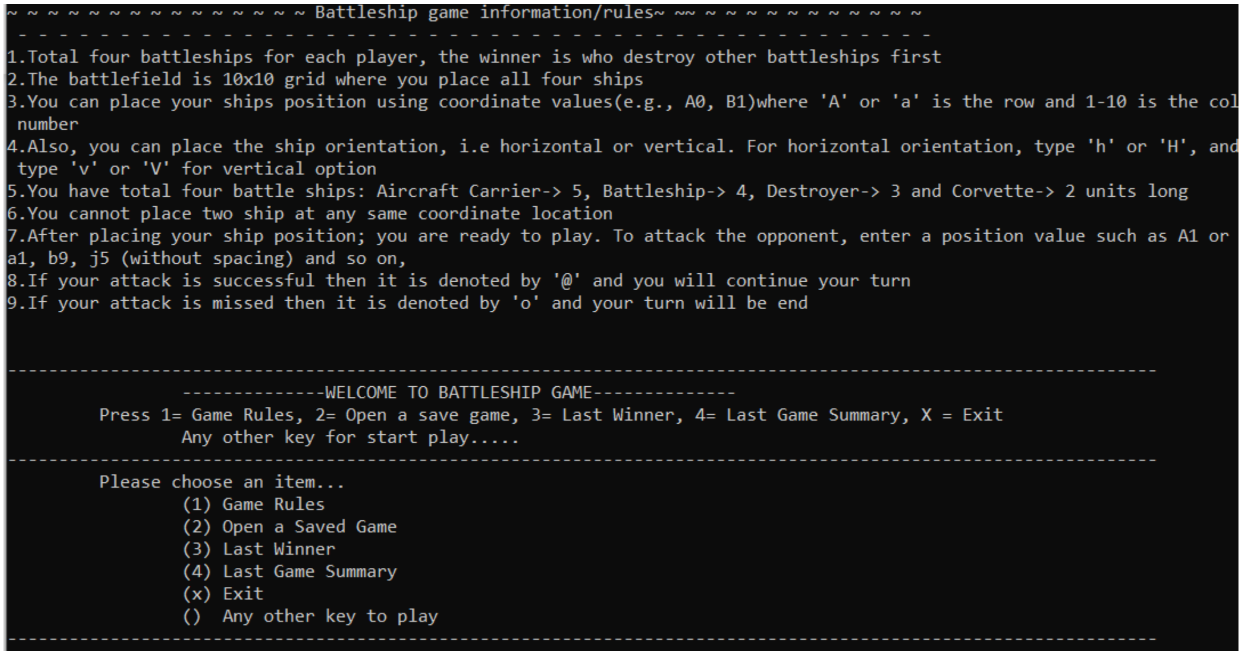
When run the game, it will display a menu like as Figure 1.



```
--------------------------------------------------------------------------------------------------------------
                 -------------WELCOME TO BATTLESHIP GAME-------------
         Press 1= Game Rules, 2= Open a save game, 3= Last Winner, 4= Last Game Summary, X = Exit
                 Any other key for start play.....
--------------------------------------------------------------------------------------------------------------
         Please choose an item...
                 (1) Game Rules
                 (2) Open a Saved Game
                 (3) Last Winner
                 (4) Last Game Summary
                 (x) Exit
                 ()  Any other key to play
--------------------------------------------------------------------------------------------------------------
```

Figure 1. Game Menu

If you press 1, then game rules will be displayed as Figure 2.



```
~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ Battleship game information/rules~ ~~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1.Total four battleships for each player, the winner is who destroy other battleships first
2.The battlefield is 10x10 grid where you place all four ships
3.You can place your ships position using coordinate values(e.g., A0, B1)where 'A' or 'a' is the row and 1-10 is the col
 number
4.Also, you can place the ship orientation, i.e horizontal or vertical. For horizontal orientation, type 'h' or 'H', and
 type 'v' or 'V' for vertical option
5.You have total four battle ships: Aircraft Carrier-> 5, Battleship-> 4, Destroyer-> 3 and Corvette-> 2 units long
6.You cannot place two ship at any same coordinate location
7.After placing your ship position; you are ready to play. To attack the opponent, enter a position value such as A1 or
a1, b9, j5 (without spacing) and so on,
8.If your attack is successful then it is denoted by '@' and you will continue your turn
9.If your attack is missed then it is denoted by 'o' and your turn will be end



--------------------------------------------------------------------------------------------------------------
                 -------------WELCOME TO BATTLESHIP GAME-------------
         Press 1= Game Rules, 2= Open a save game, 3= Last Winner, 4= Last Game Summary, X = Exit
                 Any other key for start play.....
--------------------------------------------------------------------------------------------------------------
         Please choose an item...
                 (1) Game Rules
                 (2) Open a Saved Game
                 (3) Last Winner
                 (4) Last Game Summary
                 (x) Exit
                 ()  Any other key to play
--------------------------------------------------------------------------------------------------------------
```

Figure 2. Game information/rules

If you press 2, then it will open a save game (if it available). If a save game is not available then it will display the following message:

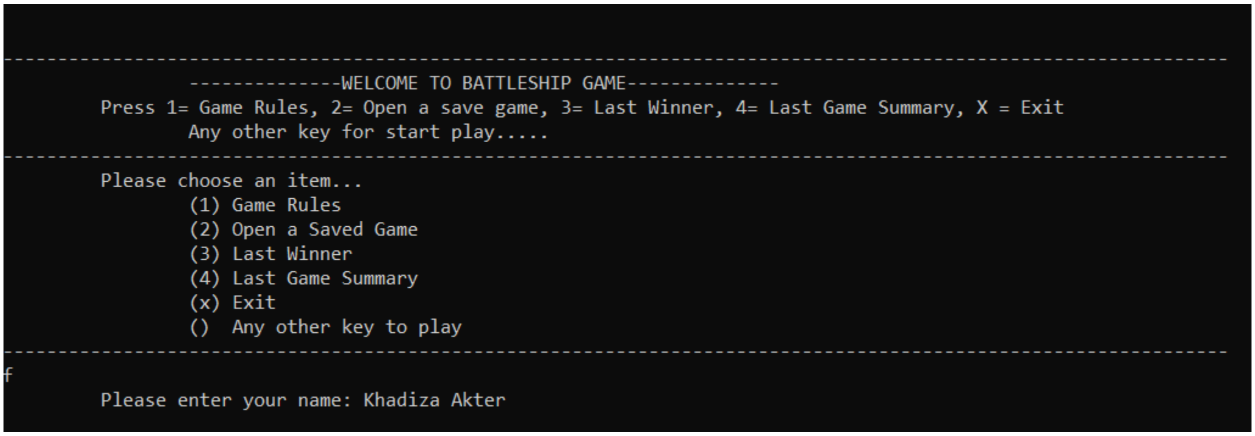
```
“Saved game is not available now”
```

If you press 3, then it will display the last winner of the game e.g., ‘human’ or ‘computer’. If it is first-game then no information available, and will display the following message:

```
"No information available now"
```

Press ‘x’ for exit the game now.

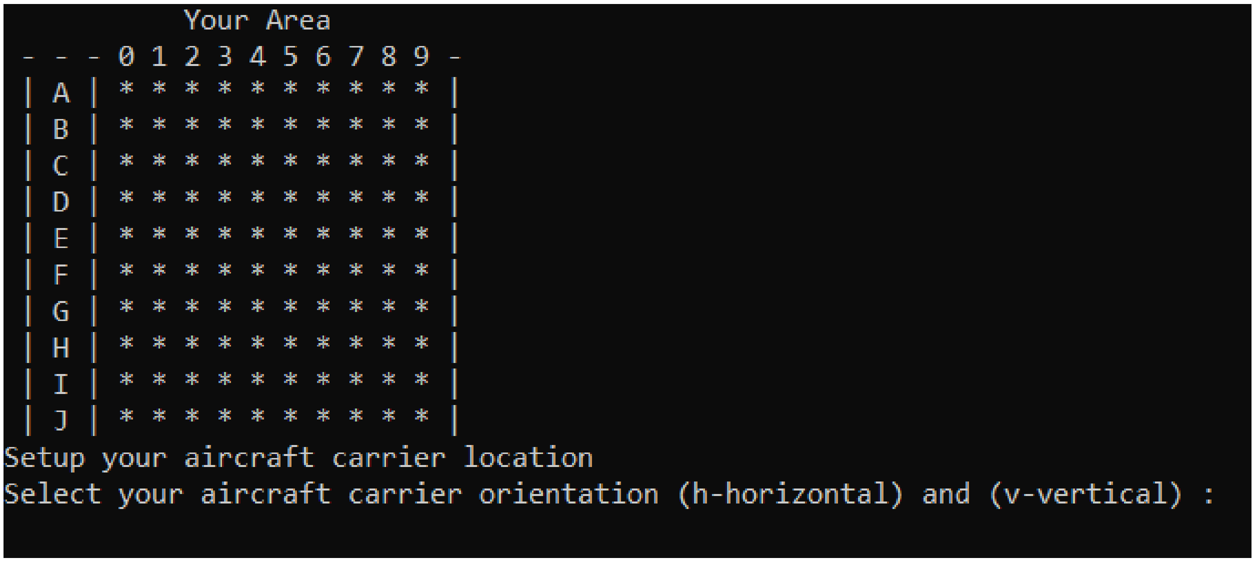
Press any other key will continue to play the game and first will ask your name, and Figure 3 shows the screen-shot of start a new game. Here player input the character ‘k’ and a new game is start and asking for player name.



```
-----------------------------------------------------------------------------------------------
              -------------WELCOME TO BATTLESHIP GAME-------------
       Press 1= Game Rules, 2= Open a save game, 3= Last Winner, 4= Last Game Summary, X = Exit
              Any other key for start play.....
-----------------------------------------------------------------------------------------------
       Please choose an item...
              (1) Game Rules
              (2) Open a Saved Game
              (3) Last Winner
              (4) Last Game Summary
              (x) Exit
              ()  Any other key to play
-----------------------------------------------------------------------------------------------
f
       Please enter your name: Khadiza Akter
```

Figure 3. A new game start

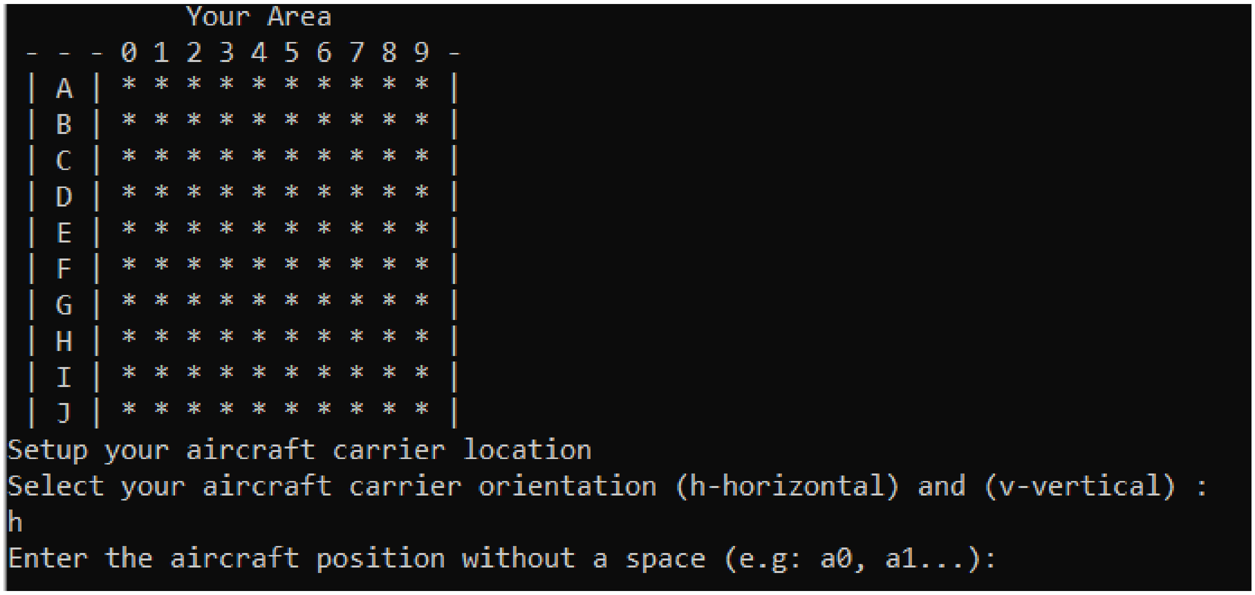
After inputting the player name, the player will see the new screen as Figure 4 where a player setup the battle ships.



```
           Your Area
 - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * * * * * * * * |
 | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |
 | D | * * * * * * * * * * |
 | E | * * * * * * * * * * |
 | F | * * * * * * * * * * |
 | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |
 | J | * * * * * * * * * * |
Setup your aircraft carrier location
Select your aircraft carrier orientation (h-horizontal) and (v-vertical) :
```

Figure 4. Setup player battle ships

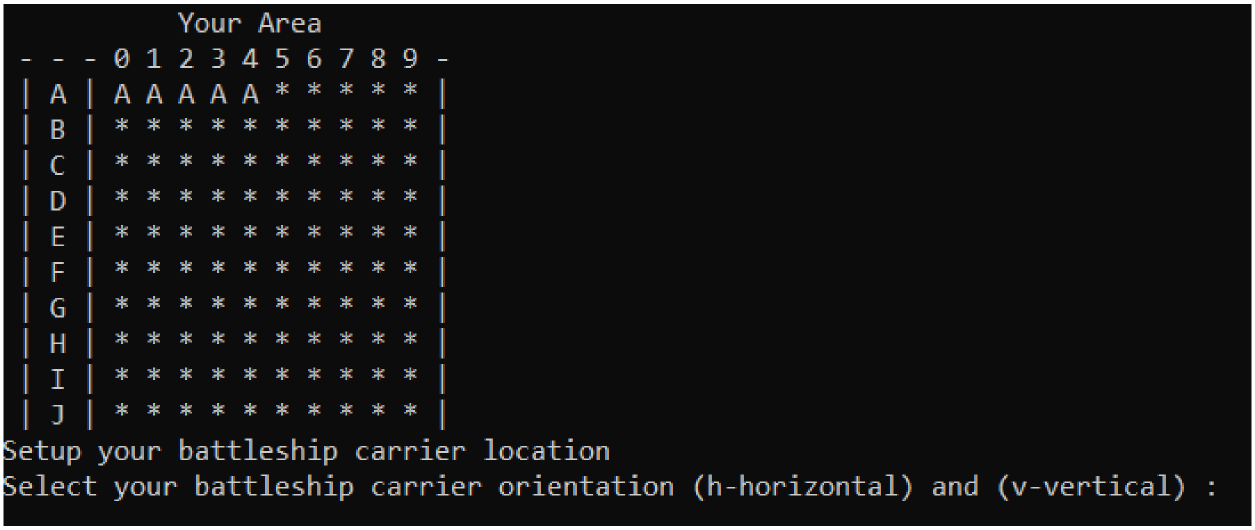
At first, player will setup the aircraft location and asking for orientation of the aircraft orientation (Figure 5). The letter ‘h’ or ‘H’ will allow for horizontal orientation, and ‘v’ or ‘V’ will allow for the vertical orientation. After choosing the orientation, player will input the starting position of the aircraft location, for example, ‘a0’ or ‘A0’, start position will be the first row and column position 0. Since aircarft size is 5 units, it will be the first row and 0, 1, 2, 3, 4 columns.



```
           Your Area
 - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * * * * * * * * |
 | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |
 | D | * * * * * * * * * * |
 | E | * * * * * * * * * * |
 | F | * * * * * * * * * * |
 | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |
 | J | * * * * * * * * * * |
Setup your aircraft carrier location
Select your aircraft carrier orientation (h-horizontal) and (v-vertical) :
h
Enter the aircraft position without a space (e.g: a0, a1...):
```

Figure 5. Aircraft setup

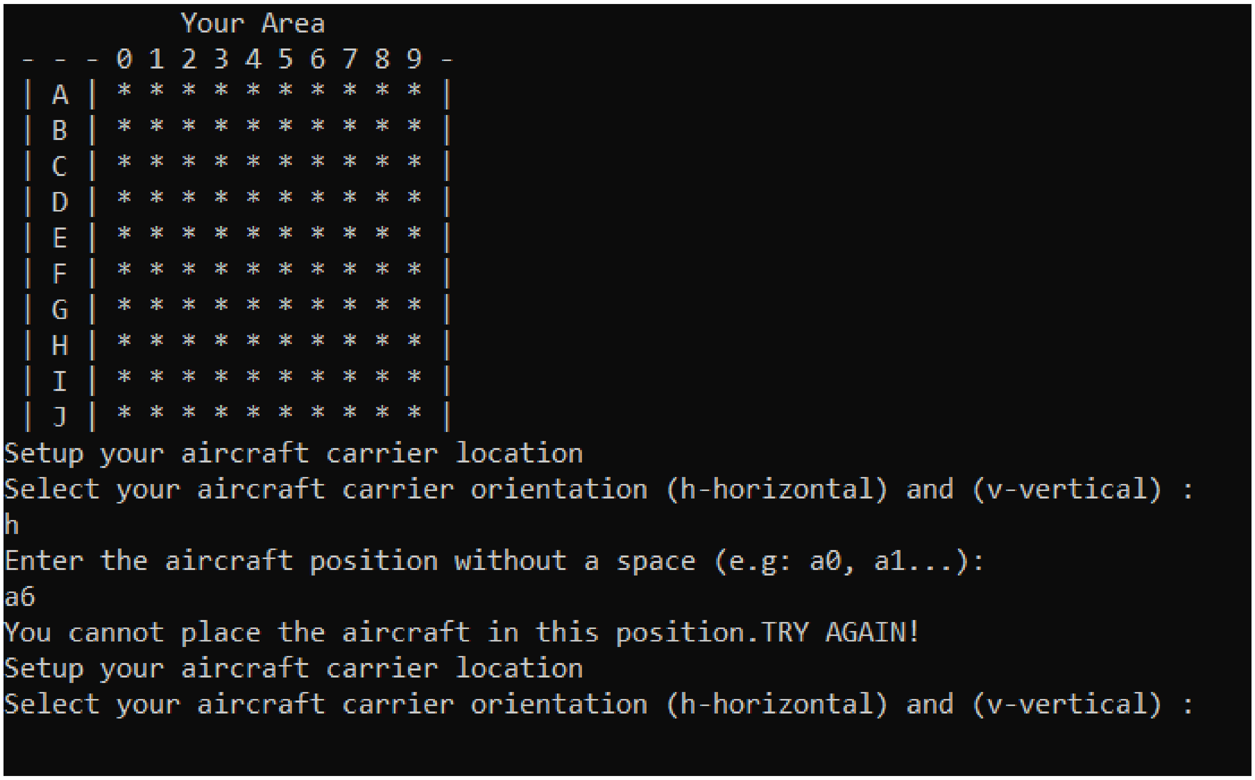
The Figure 6 is the screen when setup the aircraft orientation horizontal, ‘h’ and position ‘a0’, asking for setup the next ship position which is battleship.



```
           Your Area
 - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | A A A A A * * * * * |
 | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |
 | D | * * * * * * * * * * |
 | E | * * * * * * * * * * |
 | F | * * * * * * * * * * |
 | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |
 | J | * * * * * * * * * * |
Setup your battleship carrier location
Select your battleship carrier orientation (h-horizontal) and (v-vertical) :
```

Figure 6. Setup aircraft as 'h' and position 'a0'

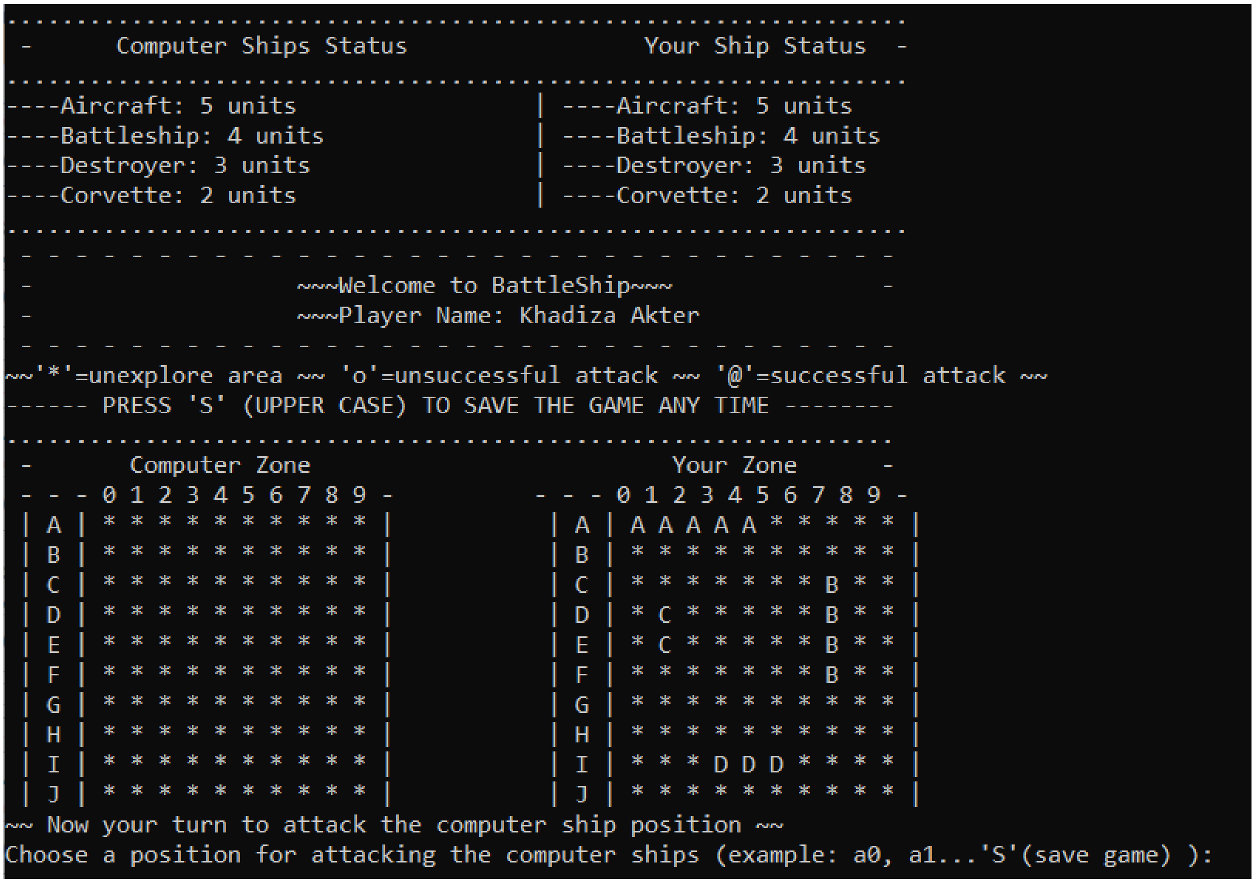
If the player choose choose an aircraft position horizontal 'h' and start position a6, b6, c6 …..j6 or more then 5 units cannot be fit with four slots of the grid and return an error message to the player as Figure 7. The player can again setup the ship position.



```
           Your Area
 - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * * * * * * * * |
 | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |
 | D | * * * * * * * * * * |
 | E | * * * * * * * * * * |
 | F | * * * * * * * * * * |
 | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |
 | J | * * * * * * * * * * |
Setup your aircraft carrier location
Select your aircraft carrier orientation (h-horizontal) and (v-vertical) :
h
Enter the aircraft position without a space (e.g: a0, a1...):
a6
You cannot place the aircraft in this position.TRY AGAIN!
Setup your aircraft carrier location
Select your aircraft carrier orientation (h-horizontal) and (v-vertical) :
```

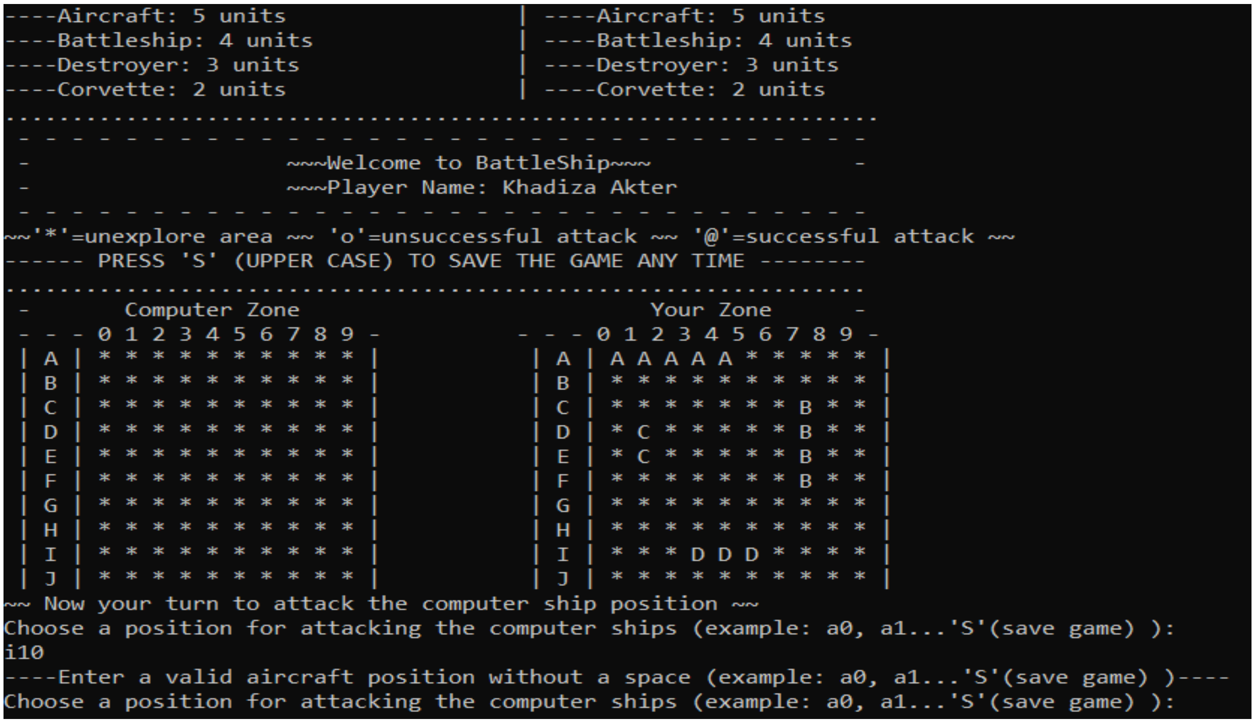
Figure 7. Setup aircraft as 'h' and position 'a6'

The player will able to setup all four vehicles according to his/her choice. As soon as, the player setups all the vehicles, computer setup it's vehicles within fraction of second, and the game playing screen will appear as Figure 8. At the top, the ship status will display for both computer and player. At the beging of play, all ships sizes are similar to the ship sizes. In the game matrix, '*' represents the unexplore area, 'o' represents the miss hit, and '@' represents the successfully hit. A player can save the game anytime by pressing the UPPER CASE 'S'. The player will choose a position for attacking the computer's ships. If the player chooses a wrong position, he/she will receive a message and will ask to input the position again. Figure 9 shows that the player inputs the i10 which position out of the bound and asking for to provide the input again.



```
.................................................................
 -      Computer Ships Status                 Your Ship Status  -
.................................................................
----Aircraft: 5 units                   | ----Aircraft: 5 units
----Battleship: 4 units                 | ----Battleship: 4 units
----Destroyer: 3 units                  | ----Destroyer: 3 units
----Corvette: 2 units                   | ----Corvette: 2 units
.................................................................
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                   ~~~Welcome to BattleShip~~~               -
 -                   ~~~Player Name: Khadiza Akter
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
................................................................
 -       Computer Zone                           Your Zone      -
 - - - 0 1 2 3 4 5 6 7 8 9 -           - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * * * * * * * * |            | A | A A A A A * * * * * |
 | B | * * * * * * * * * * |            | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |            | C | * * * * * * * B * * |
 | D | * * * * * * * * * * |            | D | * C * * * * * B * * |
 | E | * * * * * * * * * * |            | E | * C * * * * * B * * |
 | F | * * * * * * * * * * |            | F | * * * * * * * B * * |
 | G | * * * * * * * * * * |            | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |            | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |            | I | * * * D D D * * * * |
 | J | * * * * * * * * * * |            | J | * * * * * * * * * * |
~~ Now your turn to attack the computer ship position ~~
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
```

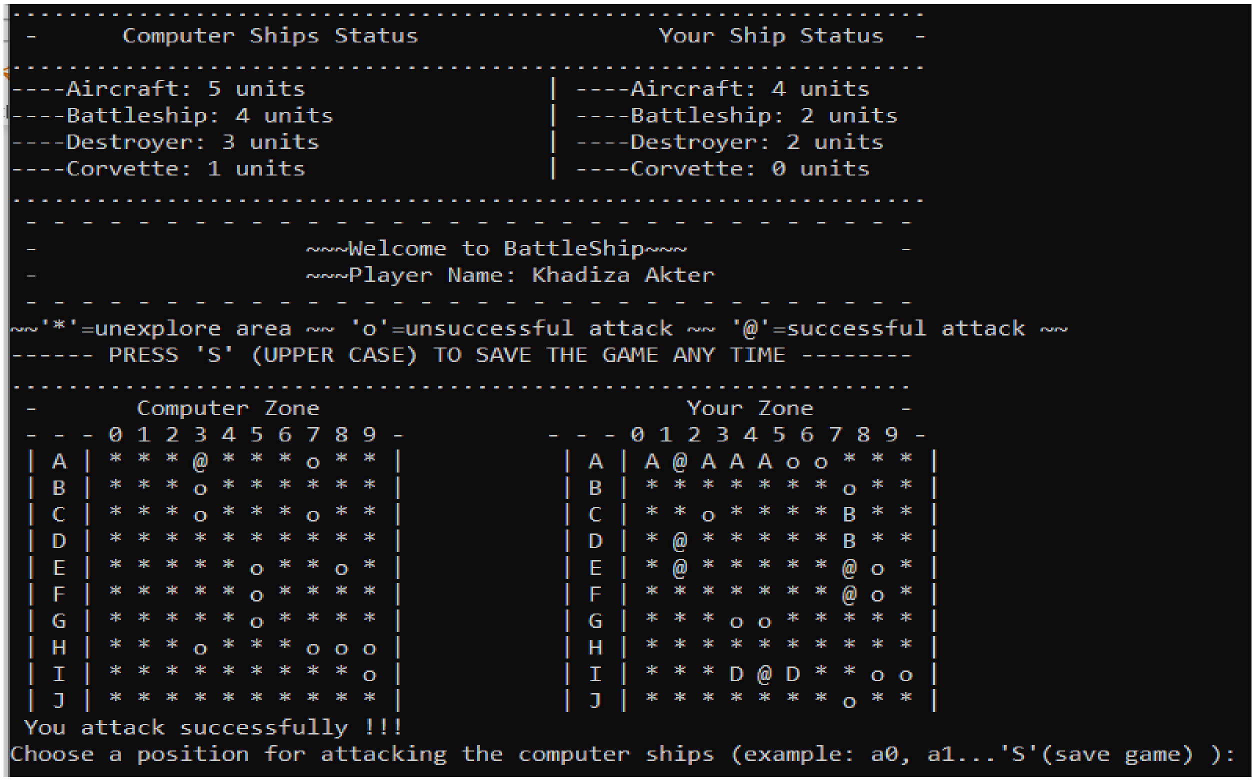
Figure 8. Game playing screen



```
----Aircraft: 5 units                   | ----Aircraft: 5 units
----Battleship: 4 units                 | ----Battleship: 4 units
----Destroyer: 3 units                  | ----Destroyer: 3 units
----Corvette: 2 units                   | ----Corvette: 2 units
.................................................................
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                   ~~~Welcome to BattleShip~~~               -
 -                   ~~~Player Name: Khadiza Akter
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
................................................................
 -       Computer Zone                           Your Zone      -
 - - - 0 1 2 3 4 5 6 7 8 9 -           - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * * * * * * * * |            | A | A A A A A * * * * * |
 | B | * * * * * * * * * * |            | B | * * * * * * * * * * |
 | C | * * * * * * * * * * |            | C | * * * * * * * B * * |
 | D | * * * * * * * * * * |            | D | * C * * * * * B * * |
 | E | * * * * * * * * * * |            | E | * C * * * * * B * * |
 | F | * * * * * * * * * * |            | F | * * * * * * * B * * |
 | G | * * * * * * * * * * |            | G | * * * * * * * * * * |
 | H | * * * * * * * * * * |            | H | * * * * * * * * * * |
 | I | * * * * * * * * * * |            | I | * * * D D D * * * * |
 | J | * * * * * * * * * * |            | J | * * * * * * * * * * |
~~ Now your turn to attack the computer ship position ~~
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
i10
----Enter a valid aircraft position without a space (example: a0, a1...'S'(save game) )----
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
```

Figure 9. Error input and asking for input position again

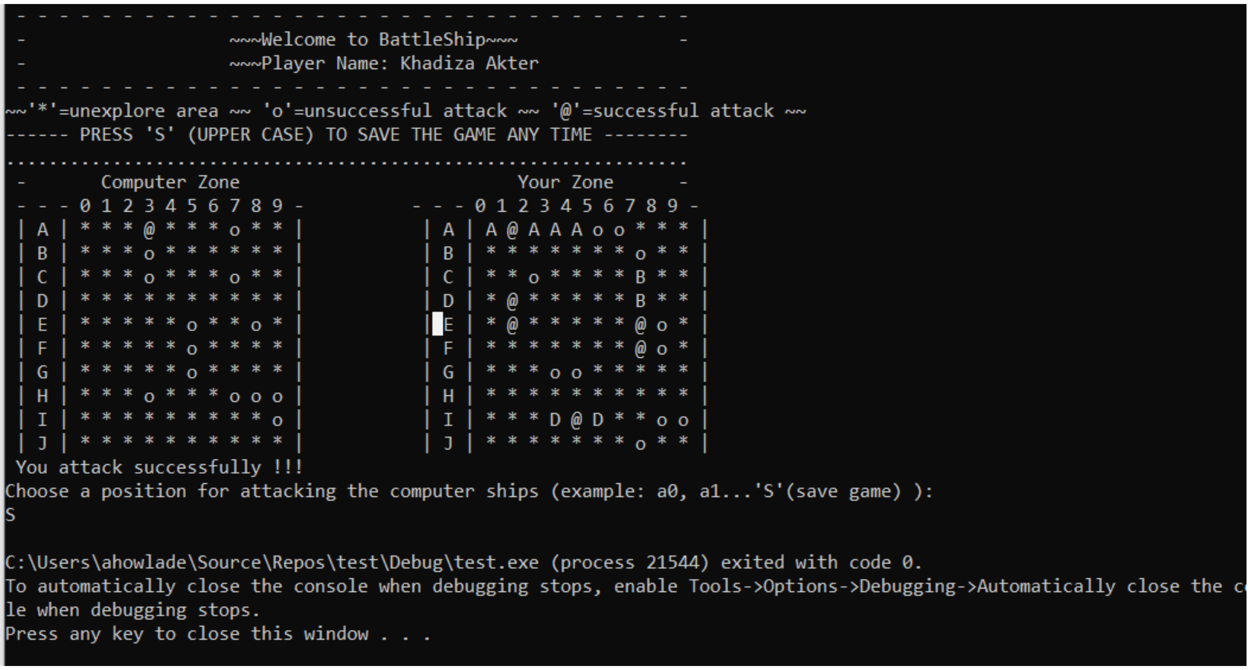
When the player or computer hits successfully then they allow to hit again. In the Figure 10, the player hits was successed and asked for inserting the position again.



```
-----------------------------------------------------------------
 -      Computer Ships Status                 Your Ship Status   -
-----------------------------------------------------------------
----Aircraft: 5 units                  | ----Aircraft: 4 units
----Battleship: 4 units                | ----Battleship: 2 units
----Destroyer: 3 units                 | ----Destroyer: 2 units
----Corvette: 1 units                  | ----Corvette: 0 units
-----------------------------------------------------------------
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                   ~~~Welcome to BattleShip~~~                -
 -                   ~~~Player Name: Khadiza Akter
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
----------------------------------------------------------------
 -       Computer Zone                       Your Zone         -
 - - - 0 1 2 3 4 5 6 7 8 9 -          - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * @ * * * o * * |            | A | A @ A A A o o * * * |
 | B | * * * o * * * * * * |            | B | * * * * * * * o * * |
 | C | * * * o * * * o * * |            | C | * * o * * * * B * * |
 | D | * * * * * * * * * * |            | D | * @ * * * * * B * * |
 | E | * * * * * o * * o * |            | E | * @ * * * * * @ o * |
 | F | * * * * * o * * * * |            | F | * * * * * * * @ o * |
 | G | * * * * * o * * * * |            | G | * * * o o * * * * * |
 | H | * * * o * * * o o o |            | H | * * * * * * * * * * |
 | I | * * * * * * * * * o |            | I | * * * D @ D * * o o |
 | J | * * * * * * * * * * |            | J | * * * * * * * o * * |
 You attack successfully !!!
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
```

Figure 10. Player attack successful and ask for position again to hit

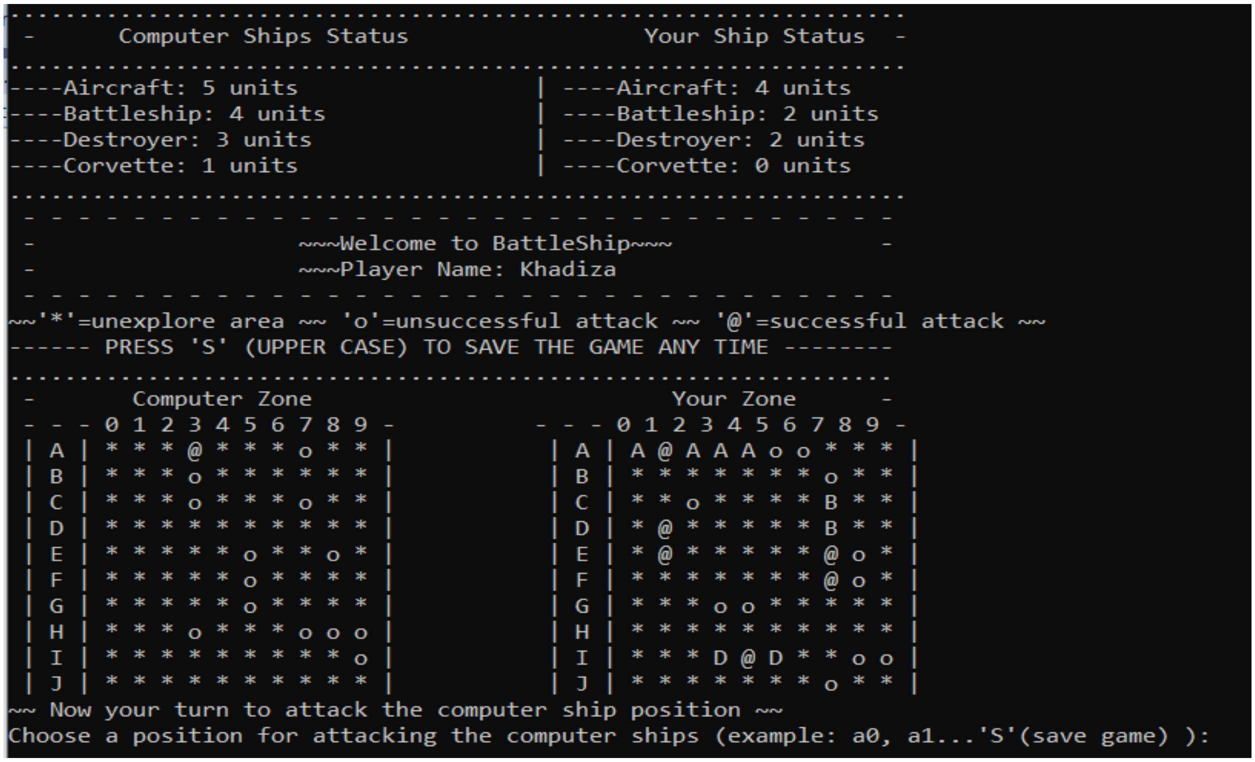
Also, at the top, the ship status was updated. Now **press ‘S’** to save the game at this stage. When press the ‘S’, the game was saved and exit. Figure 11 represents the save game and exit the program. Now run the game again and choose ‘2’ for open the save game. Figure 12 shows that the save game loaded successfully at the saving stage.



```
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                 ~~~Welcome to BattleShip~~~               -
 -                 ~~~Player Name: Khadiza Akter
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
...............................................................
 -       Computer Zone                           Your Zone      -
 - - - 0 1 2 3 4 5 6 7 8 9 -          - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * @ * * * o * * |           | A | A @ A A A o o * * * |
 | B | * * * o * * * * * * |           | B | * * * * * * * o * * |
 | C | * * * o * * * o * * |           | C | * * o * * * * B * * |
 | D | * * * * * * * * * * |           | D | * @ * * * * * B * * |
 | E | * * * * * o * * o * |           | E | * @ * * * * * @ o * |
 | F | * * * * * o * * * * |           | F | * * * * * * * @ o * |
 | G | * * * * * o * * * * |           | G | * * * o o * * * * * |
 | H | * * * o * * * o o o |           | H | * * * * * * * * * * |
 | I | * * * * * * * * * o |           | I | * * * D @ D * * o o |
 | J | * * * * * * * * * * |           | J | * * * * * * * o * * |
 You attack successfully !!!
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
S

C:\Users\ahowlade\Source\Repos\test\Debug\test.exe (process 21544) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the c
le when debugging stops.
Press any key to close this window . . .
```

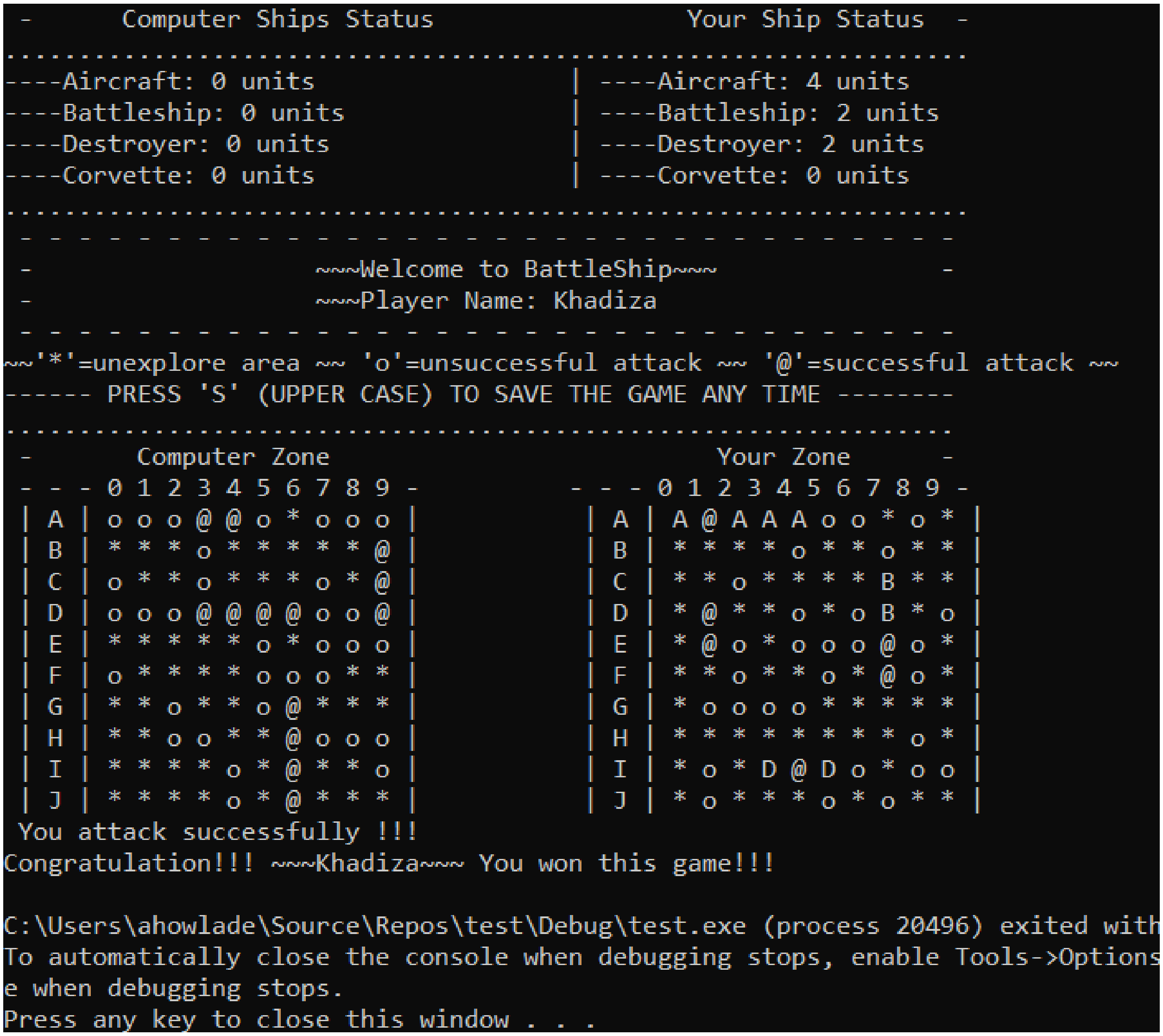
Figure 11. Save the game and exit



```
................................................................
 -      Computer Ships Status                 Your Ship Status  -
................................................................
----Aircraft: 5 units                   | ----Aircraft: 4 units
----Battleship: 4 units                 | ----Battleship: 2 units
----Destroyer: 3 units                  | ----Destroyer: 2 units
----Corvette: 1 units                   | ----Corvette: 0 units
................................................................
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                 ~~~Welcome to BattleShip~~~               -
 -                 ~~~Player Name: Khadiza                   -
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
...............................................................
 -       Computer Zone                           Your Zone      -
 - - - 0 1 2 3 4 5 6 7 8 9 -          - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | * * * @ * * * o * * |           | A | A @ A A A o o * * * |
 | B | * * * o * * * * * * |           | B | * * * * * * * o * * |
 | C | * * * o * * * o * * |           | C | * * o * * * * B * * |
 | D | * * * * * * * * * * |           | D | * @ * * * * * B * * |
 | E | * * * * * o * * o * |           | E | * @ * * * * * @ o * |
 | F | * * * * * o * * * * |           | F | * * * * * * * @ o * |
 | G | * * * * * o * * * * |           | G | * * * o o * * * * * |
 | H | * * * o * * * o o o |           | H | * * * * * * * * * * |
 | I | * * * * * * * * * o |           | I | * * * D @ D * * o o |
 | J | * * * * * * * * * * |           | J | * * * * * * * o * * |
~~ Now your turn to attack the computer ship position ~~
Choose a position for attacking the computer ships (example: a0, a1...'S'(save game) ):
```

Figure 12. Open the save game

Figure 13 represents that the player won the game and exited the game.

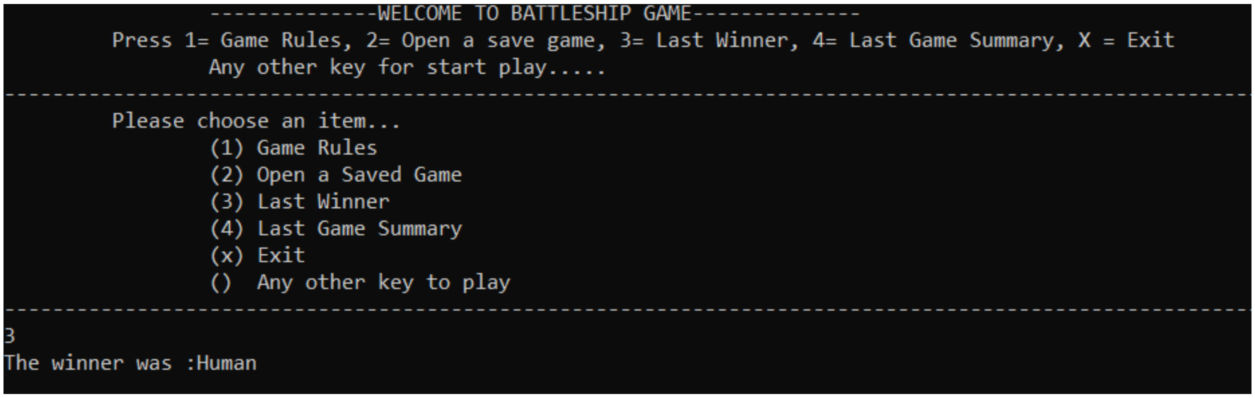


```
 -      Computer Ships Status                 Your Ship Status  -
.................................................................
----Aircraft: 0 units                   | ----Aircraft: 4 units
----Battleship: 0 units                 | ----Battleship: 2 units
----Destroyer: 0 units                  | ----Destroyer: 2 units
----Corvette: 0 units                   | ----Corvette: 0 units
.................................................................
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 -                   ~~~Welcome to BattleShip~~~               -
 -                   ~~~Player Name: Khadiza
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ '@'=successful attack ~~
------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------
................................................................
 -       Computer Zone                          Your Zone      -
 - - - 0 1 2 3 4 5 6 7 8 9 -          - - - 0 1 2 3 4 5 6 7 8 9 -
 | A | o o o @ @ o * o o o |            | A | A @ A A A o o * o * |
 | B | * * * o * * * * * @ |            | B | * * * * o * * o * * |
 | C | o * * o * * * o * @ |            | C | * * o * * * * B * * |
 | D | o o o @ @ @ @ o o @ |            | D | * @ * * o * o B * o |
 | E | * * * * * o * o o o |            | E | * @ o * o o o @ o * |
 | F | o * * * * o o o * * |            | F | * * o * * o * @ o * |
 | G | * * o * * o @ * * * |            | G | * o o o o * * * * * |
 | H | * * o o * * @ o o o |            | H | * * * * * * * * o * |
 | I | * * * * o * @ * * o |            | I | * o * D @ D o * o o |
 | J | * * * * o * @ * * * |            | J | * o * * * o * o * * |
 You attack successfully !!!
Congratulation!!! ~~~Khadiza~~~ You won this game!!!

C:\Users\ahowlade\Source\Repos\test\Debug\test.exe (process 20496) exited with
To automatically close the console when debugging stops, enable Tools->Options
e when debugging stops.
Press any key to close this window . . .
```

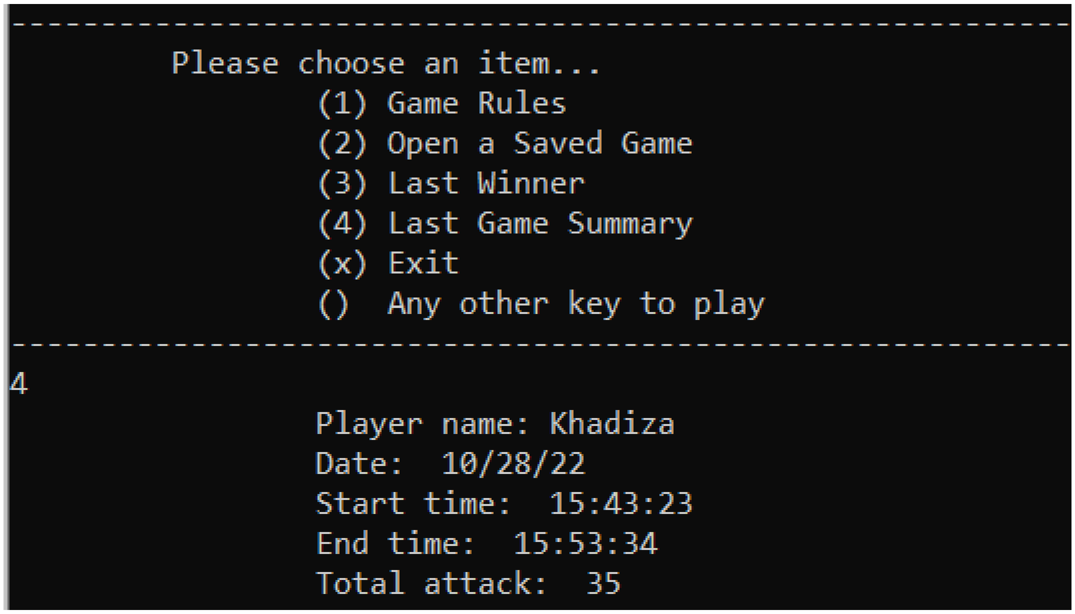
Figure 13. Player won the game and exit

The player played a game already, and if the player run the game and choose the manu item 3 as Figure 14. It will display the last winner of the game.



```
                --------------WELCOME TO BATTLESHIP GAME--------------
        Press 1= Game Rules, 2= Open a save game, 3= Last Winner, 4= Last Game Summary, X = Exit
                Any other key for start play.....
------------------------------------------------------------------------------------------------------
        Please choose an item...
                (1) Game Rules
                (2) Open a Saved Game
                (3) Last Winner
                (4) Last Game Summary
                (x) Exit
                ()  Any other key to play
------------------------------------------------------------------------------------------------------
3
The winner was :Human
```

Figure 14. Last winner of the game

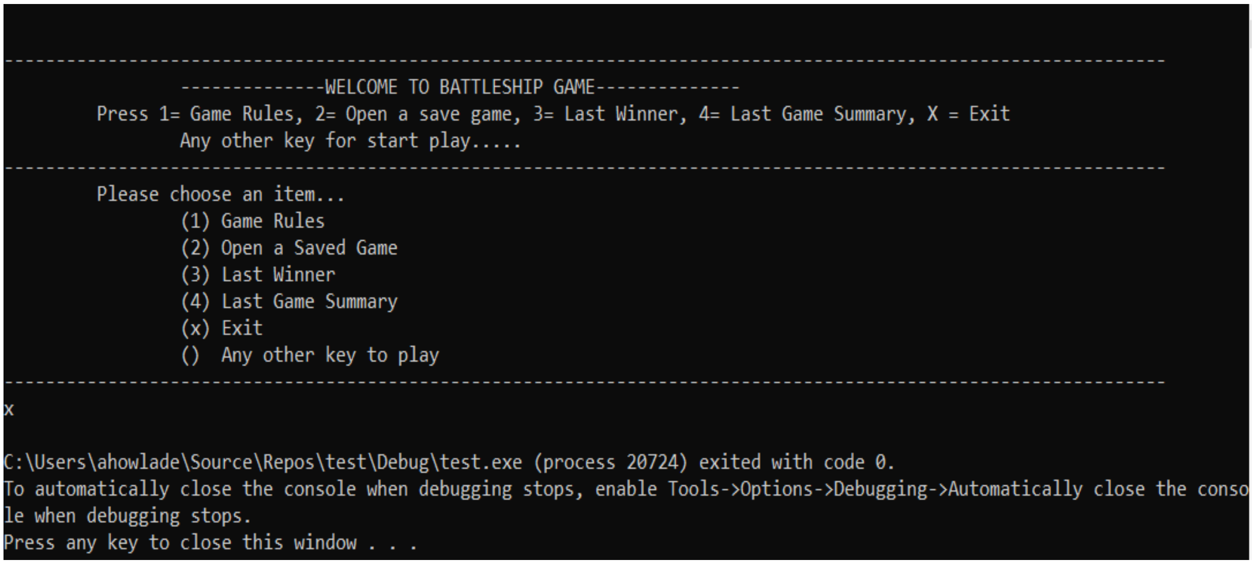


```
----------------------------------------------------------
        Please choose an item...
                (1) Game Rules
                (2) Open a Saved Game
                (3) Last Winner
                (4) Last Game Summary
                (x) Exit
                ()  Any other key to play
----------------------------------------------------------
4
                Player name: Khadiza
                Date:  10/28/22
                Start time:  15:43:23
                End time:  15:53:34
                Total attack:  35
```

Figure 15. Shows the last game summary

Figure 15 displays the game summary.

If the player chooses the menu item 2, then it will open the saved game as Figure 10 because it was last saved game. If the player press the ‘x’, the game will be exit that shows in Figure 16.



```
------------------------------------------------------------------------------------------------------------
                --------------WELCOME TO BATTLESHIP GAME--------------
        Press 1= Game Rules, 2= Open a save game, 3= Last Winner, 4= Last Game Summary, X = Exit
                Any other key for start play.....
------------------------------------------------------------------------------------------------------------
        Please choose an item...
                (1) Game Rules
                (2) Open a Saved Game
                (3) Last Winner
                (4) Last Game Summary
                (x) Exit
                ()  Any other key to play
------------------------------------------------------------------------------------------------------------
x

C:\Users\ahowlade\Source\Repos\test\Debug\test.exe (process 20724) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Figure 16. ‘x’ for exit the game

3.3 Specifications of the game

3.3.1 UML Diagram

The overall UML diagram of the project is given as follow:

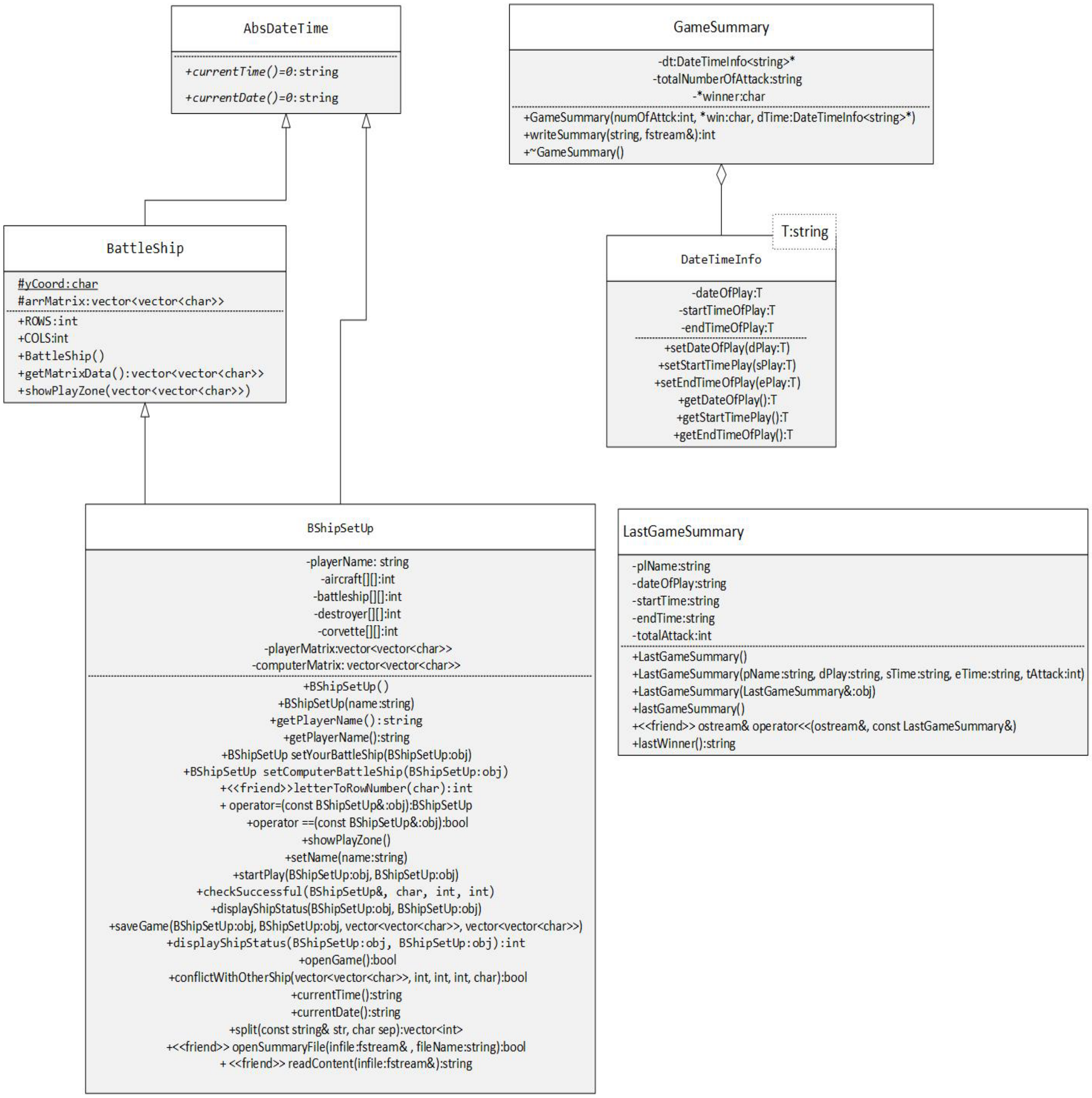

AbsDateTime
+currentTime()=0:string
+currentDate()=0:string
GameSummary
-dt:DateTimeInfo<string>*
-totalNumberOfAttack:string
-*winner:char
+GameSummary(numOfAttck:int, *win:char, dTime:DateTimeInfo<string>*)
+writeSummary(string, fstream&):int
+~GameSummary()
BattleShip
#yCoord:char
#arrMatrix:vector<vector<char>>
+ROWS:int
+COLS:int
+BattleShip()
+getMatrixData():vector<vector<char>>
+showPlayZone(vector<vector<char>>)
T:string
DateTimeInfo
-dateOfPlay:T
-startTimeOfPlay:T
-endTimeOfPlay:T
+setDateOfPlay(dPlay:T)
+setStartTimePlay(sPlay:T)
+setEndTimeOfPlay(ePlay:T)
+getDateOfPlay():T
+getStartTimePlay():T
+getEndTimeOfPlay():T
BShipSetUp
-playerName: string
-aircraft[][]:int
-battleship[][]:int
-destroyer[][]:int
-corvette[][]:int
-playerMatrix:vector<vector<char>>
-computerMatrix: vector<vector<char>>
+BShipSetUp()
+BShipSetUp(name:string)
+getPlayerName():string
+getPlayerName():string
+BShipSetUp setYourBattleShip(BShipSetUp:obj)
+BShipSetUp setComputerBattleShip(BShipSetUp:obj)
+<<friend>>letterToRowNumber(char):int
+ operator=(const BShipSetUp&:obj):BShipSetUp
+operator ==(const BShipSetUp&:obj):bool
+showPlayZone()
+setName(name:string)
+startPlay(BShipSetUp:obj, BShipSetUp:obj)
+checkSuccessful(BShipSetUp&, char, int, int)
+displayShipStatus(BShipSetUp:obj, BShipSetUp:obj)
+saveGame(BShipSetUp:obj, BShipSetUp:obj, vector<vector<char>>, vector<vector<char>>)
+displayShipStatus(BShipSetUp:obj, BShipSetUp:obj):int
+openGame():bool
+conflictWithOtherShip(vector<vector<char>>, int, int, int, char):bool
+currentTime():string
+currentDate():string
+split(const string& str, char sep):vector<int>
+<<friend>> openSummaryFile(infile:fstream& , fileName:string):bool
+ <<friend>> readContent(infile:fstream&):string
LastGameSummary
-plName:string
-dateOfPlay:string
-startTime:string
-endTime:string
-totalAttack:int
+LastGameSummary()
+LastGameSummary(pName:string, dPlay:string, sTime:string, eTime:string, tAttack:int)
+LastGameSummary(LastGameSummary&:obj)
+lastGameSummary()
+<<friend>> ostream& operator<<(ostream&, const LastGameSummary&)
+lastWinner():string

Figure 17. UML Diagram of the game

3.3.2 Method Caller Diagram

A method caller diagram of the project is given in below. The caller diagram shows a hierarchy of the methods call from the main method.

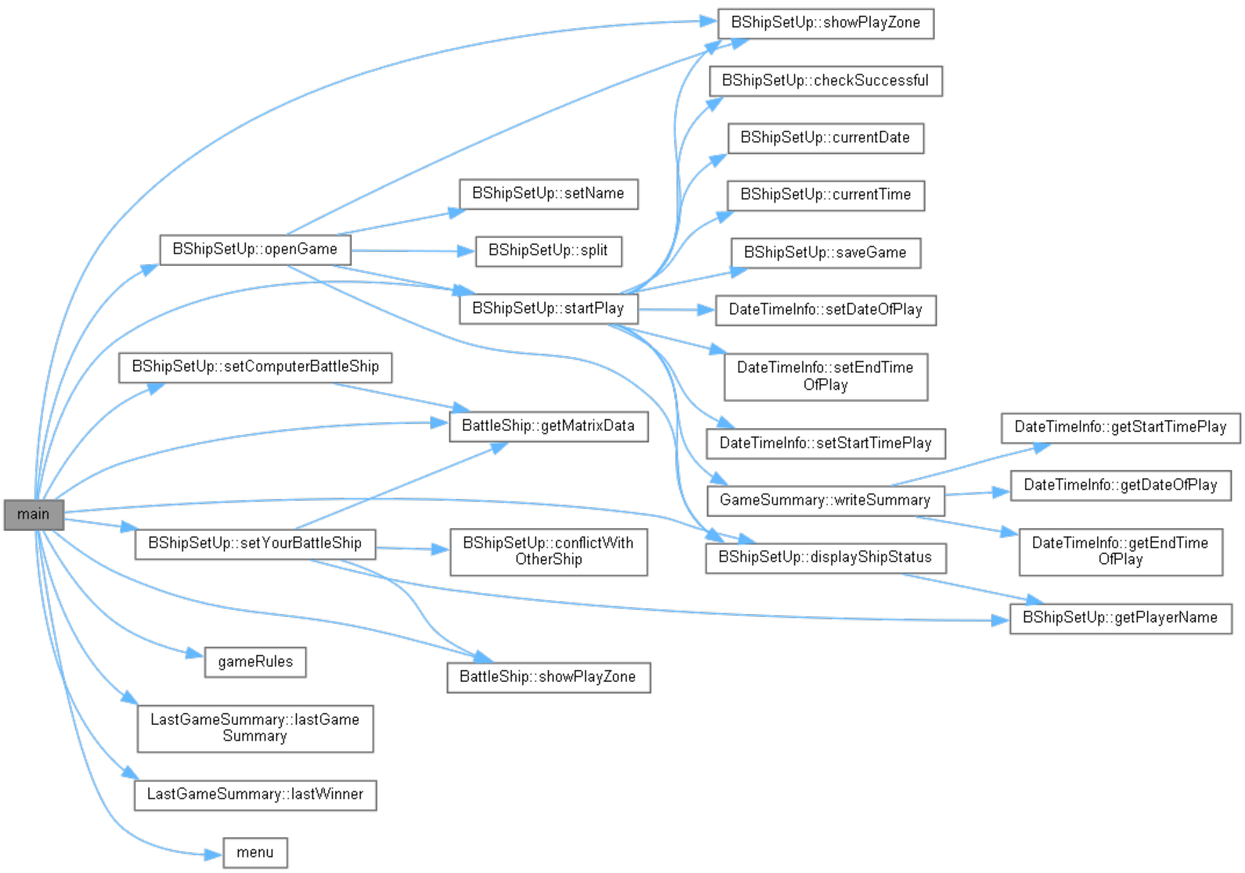
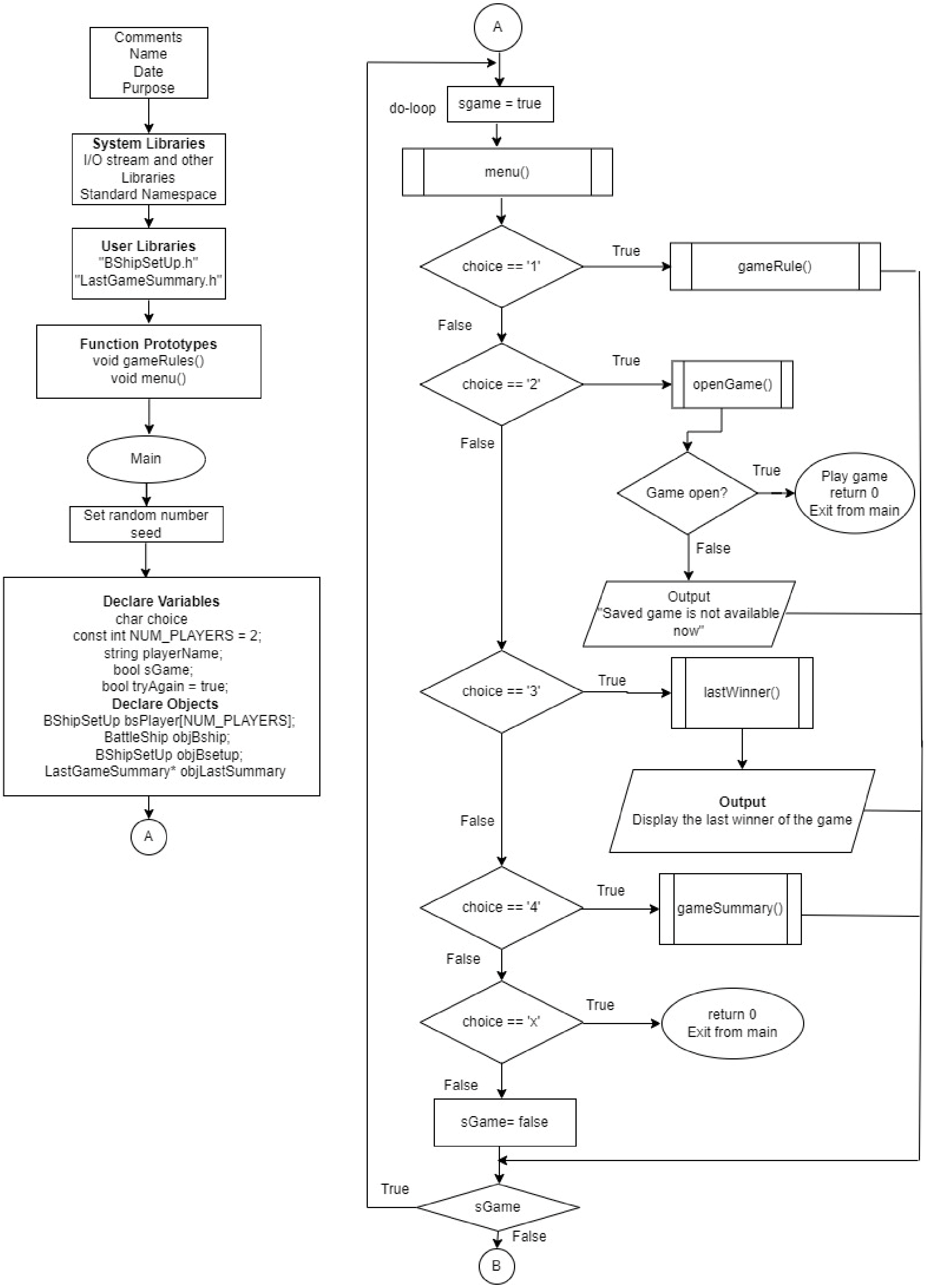

BShipSetUp::showPlayZone
BShipSetUp::checkSuccessful
BShipSetUp::currentDate
BShipSetUp::setName
BShipSetUp::currentTime
BShipSetUp::openGame
BShipSetUp::split
BShipSetUp::saveGame
BShipSetUp::startPlay
DateTimeInfo::setDateOfPlay
BShipSetUp::setComputerBattleShip
DateTimeInfo::setEndTime OfPlay
BattleShip::getMatrixData
DateTimeInfo::setStartTimePlay
DateTimeInfo::getStartTimePlay
GameSummary::writeSummary
DateTimeInfo::getDateOfPlay
main
BShipSetUp::setYourBattleShip
BShipSetUp::conflictWith OtherShip
BShipSetUp::displayShipStatus
DateTimeInfo::getEndTime OfPlay
BShipSetUp::getPlayerName
gameRules
BattleShip::showPlayZone
LastGameSummary::lastGame Summary
LastGameSummary::lastWinner
menu

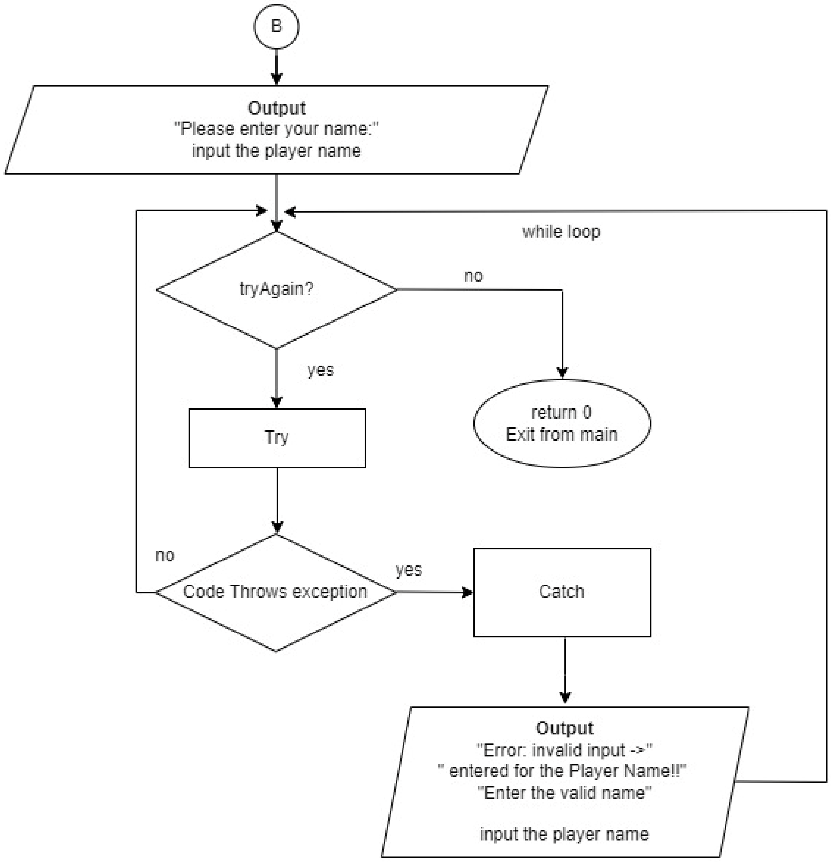
Figure 18. A method caller diagram of the project

3.3.1 Flowchart

The flow diagram of the game is shown in below:



Flowchart (continue)



Flowchart of the game

3.3.2 Pseudocode

The pseudocode of the program is shown in below:

```
Create the object for BattleShip, BShipSetUp and LastGameSummary
        repeat
            set sGame = true
            draw the menu
            choice a menu item
            if choice is 1 then
                Show the game rules
            else if choice is 2 then
                Open the save game
            else if choice is 3 then
            Display the last winner of the game
        else if choice is 4 then
            Show the last game summary
        else if choice is ‘x’ then
            Exit the game
        else
            Set sGame = false
        end if
    Until sGame is false
    Input the player name
    While tryAgain do
        Try
            Setup player name using BShipSetUp constructor
            Draw the player zone only and set the battleships
            Randomly computer’s set the battleships
            Display the status of ships both player and computer
            Draw the both player and computer zones
            Start the play, and save, win, or loss the game
            Set tryAgain = false
        Catch Exception
    Exit the game
```

3.4 Concept Used

Chapter 13

13.3 Defining an Instance of a Class
13.5 Focus on Software Engineering
13.6 Inline member functions
13.7 Constructor
13.8 Passing Arguments to Constructors
13.9 Destructors
13.10 Overloading Constructors
13.12 Arrays of Objects
13.15 Focus on Object Oriented design:UML

Chapter 14
14.1 Instance and Static Members
14.2 Friends of Classes
14.4 Copy Constructor
14.5 Operator Overloading
14.7 Aggregation
Chapter 15
15.2 Protected member and class access
15.3 Constructors and Destructors Base and Derived classes
15.5 Class Hierarchies
15.6 Polymorphism and virtual member functions
15.7 Structures as Function Arguments
15.8 Multiple Inheritance

3.5 Major variables

Type	Name	Description	Location
Integer	ROWS	Constant rows for 10x10 matrix	BattleShip Class
	COLS	Constant column for 10x10 matrix	BattleShip Class
	NUM_PLAYERS	Number of player	main()
	AIRCRAFT_LENGTH	Unit length of the aircraft	setYourBattleShip()
	BATTLESHIP_LENGTH	Unit length of the battleship	setYourBattleShip()
	DESTROYER_LENGTH	Unit length of the destroyer	setYourBattleShip()
	CORVETTE_LENGTH	Unit length of the corvette	setYourBattleShip()
char	choice	Take input for select the menu item	main()
	shipOrientiation	Take the input for ship orientation (h or v)	setYourBattleShip()
	ch	To get a char from file	lastGameSummary()
char array	playerName	Declare the input array to take player name	main()
	yCoord	declare a char array to maintain y-coordinate	showPlayZone()
	pMat	To store player matrix save data	saveGame()
	cMat	To store computer matrix save data	saveGame()
Int arry	aircraft[5][2]	Aircraft length is 5 for tracking its coordinate value	BShipSetUp
	battleship[4][2]	Battleship length is 4 for tracking its coordinate values (row,col)	BShipSetUp
	destroyer[3][2]	Destroyer length is 3 for its coordinate values (row,col)	BShipSetUp
	corvette[2][2]	Corvette length is 2 for tracking its coordinate values (row,col)	BShipSetUp
string	playerName	Declare the input array to take player name	BShipSetUp
bool	sGame	To track do while-loop for menu	main()
	isSuccessful	Successfully attack or not	startPlay()
vector	playerMatrix	To hold player matrix information	BShipSetUp
	computerMatrix	To hold player matrix information	BShipSetUp

REFERENCES:

Learn the battleship game:

[1] Battleship (game), https://en.wikipedia.org/wiki/Battleship_(game)

Textbook for developing the project:

[2] Tony Gaddis, Starting with C++ from Control Structures Through Objects

In the developing phase, when I received an error or get a clearer concept regarding any issues; I searched it on the google and most of the cases I found the solution at the following references:

[3] https://stackoverflow.com/

Draw the diagram

[4] https://doxygen.nl/

Program Listing:

main.cpp

```
/*
 * File:   main.cpp
 * Author: Khadiza Akter
 * Created on December 05, 2022, 7:57 PM
 * Purpose: Game of BattleShip
 *          Fill 2-D vector with '*'
 *          Apply the Rules of Battleship
 *          Set up player battleship
 *          Set up computer battleship
 *          Draw the both matrix and playing zone
 *          Start the play, allow user to input and computer to select input randomly
 *          Check the input of both player and computer that hit successfully or
           not, update both player matrix and ship status structure.
 *          Draw the number of remaining ship status both player and computer
 *          Winner of the game.
 *          Save game and game summary; open a save game and play it,
 *          update the main menu, project testing, update comments.
 *          Specification and implementation of base,derived,
           abstract(only specification), template or other classes.

 */
 //System Level Libraries
#include <iostream> //Input-output library
#include <string>   //Needed for strings
```

```
#include <cstdlib>  //Srand to set the seed
#include <ctime>    //set for time()
#include <iomanip>  //Format the output
#include <stdlib.h> //System()
using namespace std;//Standard Name-space under which System Libraries reside

//User defined libraries
#include "BShipSetUp.h" //needed for BShipSetUp class
#include "LastGameSummary.h" //needed for LastGameSummary class

//Function Prototypes
void gameRules(); //display the game rules
void menu(); //Display the menu

//Execution begins here!
int main() {
    //Random seed here
    srand(static_cast<unsigned int>(time(0)));
    //Declare variables
    char choice;       // take input for checking the start play or game rules
    const int NUM_PLAYERS = 2; // Number of player
    BShipSetUp bsPlayer[NUM_PLAYERS];// Array of class BShipSetUp
    string playerName;   //To hold the player name
    bool sGame;          // To track do while-loop for menu
    bool tryAgain = true; //Flag to reread the input for player name

    LastGameSummary *objLastSummary = new LastGameSummary; // Define a object for LastGameSummary class
    BattleShip objBship; // Define a object for BattleShip class
    BShipSetUp objBsetup; // Define a object for BShipSetUp class
    //Output the game statistics or menu to the screen
    do {
        sGame = true;
        menu();
        cin >> choice;  // Ask for input to see the rules or continue to game
        if (choice == '1')  gameRules(); //Call function to view the games rules
        else if (choice == '2') {
            bool ga = objBsetup.openGame();   // Open a save game
            if (!ga) cout << "Saved game is not available now" << endl;
            else      return 0;       // If open successfully, then play and exit
        } //End else-if
        else if (choice == '3') {
            string w = objLastSummary->lastWinner();// Display the last time winner of the match
            cout << "The winner was :" + w << endl;
        } //End else-if
        else if (choice == '4') objLastSummary->lastGameSummary();  // Display the game summary
        else if (choice == 'x') return 0;           // exit the program
        else                    sGame = false;      // Start to play
    } while (sGame); //End do-while loop

    cin.ignore(); //To ignore one or more characters from the input buffer
    cout << "\t Please enter your name: "; //Ask user to enter name
    getline(cin, playerName); // Take the player name
    while (tryAgain) {
        try {
            //create an object of BShipSetUp class and using constructor to initialize member
            BShipSetUp bSetUp(playerName);
            // Method call from the base class object that
            // will show a compile-time polymorphism
            objBship.showPlayZone(objBship.getMatrixData());
            //Set player ship position and return structure
            bsPlayer[0] = bSetUp.setYourBattleShip(bSetUp);
            //Set computer ship position and return structure
            bsPlayer[1] = bSetUp.setComputerBattleShip(bSetUp);
            int retInit = bSetUp.displayShipStatus(bsPlayer[0], bsPlayer[1]);
            //Method call from derived class that show compile-time polymorphism
            bSetUp.showPlayZone();
            bSetUp.startPlay(bsPlayer[0], bsPlayer[1]);
            tryAgain = false;
        } //End try
        catch (BShipSetUp::InvalidName) {
            cout << "\t Error: invalid input -> [" << playerName
                << "] entered for the Player Name!!" << endl;
            cout << "\t Enter the valid name: ";
            getline(cin, playerName);
        } //End catch

    } //End while-loop

    //Exit the program
    return 0;
} //end of main function

//************************************************************
//Definition of menu.
//Input->: None, data on menu item
//Output->:No return, This display menu item
//************************************************************
void menu() {
    //Display menu
    cout << endl;
    cout << endl;
    cout << setfill('-') << setw(112) << "" << endl;
    cout << "\t\t --------------WELCOME TO BATTLESHIP GAME--------------\n";
    cout << "\t Press 1= Game Rules, 2= Open a save game,";
    cout << " 3= Last Winner, 4= Last Game Summary, X = Exit" << endl;
    cout << "\t\t Any other key for start play....." << endl;
    cout << setfill('-') << setw(112) << "" << endl;
    cout << "\t Please choose an item..." << endl;
    cout << "\t\t (1) Game Rules " << endl;
    cout << "\t\t (2) Open a Saved Game " << endl;
    cout << "\t\t (3) Last Winner " << endl;
    cout << "\t\t (4) Last Game Summary " << endl;
    cout << "\t\t (x) Exit " << endl;
    cout << "\t\t ()  Any other key to play " << endl;
    cout << setfill('-') << setw(112) << "" << endl;
} //End menu function

//****************************************************************
```

```
//Definition of function gameRules
//Input->: None, data on game rules
//Output->:No return, Display the game rules
//****************************************************************
void gameRules() {

    system("cls"); //clear the screen
    cout << " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -"
        " - - - - - - - - - - - - - " << endl;
    cout << " - - - - -                        ~~~Welcome to BattleShip Game~~~    "
        "             - - - - - -" << endl;
    cout << "~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ Battleship game information/rules~  ~"
        "~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~" << endl;
    cout << " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -"
        " - - - - - - - - - - - - - -" << endl;
    cout << "1.Total four battleships for each player, the winner is who"
        " destroy other battleships first" << endl;
    cout << "2.The battlefield is 10x10 grid where you place all four ships\n";
    cout << "3.You can place your ships position using coordinate values(e.g."
        ", A0, B1)where 'A' or 'a' is the row and 1-10 is the col number\n";
    cout << "4.Also, you can place the ship orientation, i.e horizontal or "
        "vertical. For horizontal orientation, type 'h' or 'H', and type"
        " 'v' or 'V' for vertical option" << endl;
    cout << "5.You have total four battle ships: Aircraft Carrier-> 5, "
        "Battleship-> 4, Destroyer-> 3 and Corvette-> 2 units long" << endl;
    cout << "6.You cannot place two ship at any same coordinate location\n";
    cout << "7.After placing your ship position; you are ready to play. To "
        "attack the opponent, enter a position value such as A1 or a1, b9,"
        " j5 (without spacing) and so on,  " << endl;
    cout << "8.If your attack is successful then it is denoted by '@' "
        "and you will continue your turn" << endl;
    cout << "9.If your attack is missed then it is denoted by 'o'"
        " and your turn will be end" << endl;

} //End gameRules function
```

BattleShip.cpp

```
/*
 * File:   BattleShip.cpp
 * Author: Khadiza Akter
 * Created on December 05, 2022, 6:40 PM
 * Purpose: Implementation file for base class BattleShip
 */
#include <iostream> //Input-output library
#include <stdlib.h> //System()
using namespace std;//Standard Name-space under which System Libraries reside

#include "BattleShip.h"

//Definition of static member variable outside of the class
char BattleShip::yCoord[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G',
                              'H', 'I', 'J', '\0' };
```

```
    //******************************************************************
    //Definition of function drawPlayerArea.Use 2d char array as @param                              *
    //Display player matrix                                               *
    //******************************************************************
    void BattleShip::showPlayZone(vector<vector<char>> matrixData) {
        system("cls");
        int i = 0;
        // write a text head for computer zone
        // display the x-coordinate value
        cout << "           Your Area                           " << endl;
        cout << " - - - 0 1 2 3 4 5 6 7 8 9 -           " << endl;
        // loop to draw the 2-D vector using iterator vector
        for (auto it = matrixData.begin(); it != matrixData.end(); it++) {
            cout << " | " << yCoord[i] << " | ";
            for (auto element : *it) {
                cout << element << " ";
            }
            cout << "|" << endl;
            i++;
        }
}
```

BShipSetUp.cpp

```
    /*
     * File:   BShipSetUp.cpp
     * Author: Khadiza Akter
     * Created on December 5, 2022, 6:50 PM
     * Purpose: Implementation file of derived BShipSetUp class from both BattleShip
                and AbsDateTime class
     */

    #include <iostream> //Input-output library
    #include <cstdlib>  //Srand to set the seed
    #include <ctime>    //set for time()
    #include <cstring>  //For memcpy()
    #include <fstream>  //File I/O
    #include <stdlib.h> //System()
    #include <sstream>
    using namespace std;//Standard Name-space under which System Libraries reside

    #include "BShipSetUp.h"

    //******************************************************************
    //Definition of function operator==. This is an operator== overloaded
    //function of BShipSetUp class, use constant object of BShipSetUp
    //class as @param and also return boolean
    //******************************************************************
    bool BShipSetUp::operator ==(const BShipSetUp& equal) {
        if (sizeof(aircraft) == sizeof(equal.aircraft) && sizeof(battleship)
            == sizeof(equal.battleship) && sizeof(destroyer) == sizeof(equal.destroyer)
            && sizeof(corvette) == sizeof(equal.corvette)) {
            return true; // check the equality of both player and computer battle ship
        }
        else {
            return false;
        }
    }
    //***************************************************************
    //Definition of function operator=. This is an operator= overloaded
    //function of BShipSetUp class, use constant object of BShipSetUp
    //class as @param and also return constant object of that class
    //***************************************************************
    const BShipSetUp BShipSetUp::operator=(const BShipSetUp& right) {
        if (this != &right) {
            memcpy(aircraft, right.aircraft, 5 * 2 * sizeof(int));
            memcpy(battleship, right.battleship, 4 * 2 * sizeof(int));
            memcpy(destroyer, right.destroyer, 3 * 2 * sizeof(int));
            memcpy(corvette, right.corvette, 2 * 2 * sizeof(int));
        }
        return *this;
    }
    //********************************************************************
    //Definition of function letterToRowNumber.This function is declared as a                    *
    //friend by BShipSetUp class and it will determine the letter (A, B, C...J)
    // value to integer y-axis value (0,1,2...9)
    //********************************************************************
    int letterToRowNumber(char letter)
    {
        switch (letter)
        {
        case 'A':
            return 0;                       // return index 0
        case 'B':
            return 1;                       // return index 1
        case 'C':
            return 2;                       // return index 2
        case 'D':
            return 3;                       // return index 3
        case 'E':
            return 4;                       // return index 4
        case 'F':
            return 5;                       // return index 5
        case 'G':
            return 6;                       // return index 6
        case 'H':
            return 7;                       // return index 7
        case 'I':
            return 8;                       // return index 8
        case 'J':
            return 9;                       // return index 9
        }
    }
    //*************************************************************
    //Definition of function conflictWithOtherShip                    *
    //This check the ship position conflict with other ship or not    *
    //*************************************************************
```

```
bool BShipSetUp::conflictWithOtherShip(vector<vector<char>> playerMatrix,
    int row, int col, int shipLength, char shipOrientation) {
    if (shipOrientation == 'h') { // check the ship orientation
        // for horizontal orientation check the column till ship length
        for (int i = col; i < col + shipLength; i++)
        {    // check the character for position of the matrix, if it is
            // not '*' that means it is conflict with other ship position
            if (playerMatrix[row][i] != '*') {
                return true;// and return true
            }
        }
    } //end if
    else {
        // for horizontal orientation check the row till ship length
        for (int i = row; i < row + shipLength; i++)
        {
            if (playerMatrix[i][col] != '*') {
                return true;        // and return true
            }
        }
    } //end else
    return false;
}

//******************************************************************
//Definition of function setYourBattleShip                         *
//Set player ship position and return player ship position         *
//******************************************************************
BShipSetUp BShipSetUp::setYourBattleShip(BShipSetUp b) {

    const int AIRCRAFT_LENGTH = 5;     // Unit length of the aircraft
    const int BATTLESHIP_LENGTH = 4;   // Unit length of the battleship
    const int DESTROYER_LENGTH = 3;    // Unit length of the destroyer
    const int CORVETTE_LENGTH = 2;     // Unit length of the corvette
    const int POSITION_LENGTH = 2;     // Input length of a grid position
    char shipOrientiation;             // Take the input for ship orientation (h or v)
    string shipPosition = "";          // Take the input for ship starting position (a0, a2...j9 so on)
    b.playerName = getPlayerName();    // Get the player name
    playerMatrix = getMatrixData();    // Get the matrix data
    BattleShip objBShip;               // Define a battleship object

    while (true) {  // Loop for setup the aircraft position
        cout << "Setup your aircraft carrier location" << endl;
        cout << "Select your aircraft carrier orientation "
            "(h-horizontal) and (v-vertical) : " << endl;
        while (true) { // take a infinite loop for satisfying the valid input for ship orientation
            cin >> shipOrientiation;   // take the input of ship orientation 'h' or 'v'
            // compare the ship orientation input if it is 'v' or 'h' then fine
            if (tolower(shipOrientiation) == 'h'
                || tolower(shipOrientiation) == 'v') {
                cin.ignore();
                break;
            }
            else {// if input is not h or v then ask for input again
                cout << "Please enter a valid input 'H' or 'h' or 'V' or 'v' \n";
                cin.ignore();
                continue;
            } //end else
        } //end while-loop

        cout << "Enter the aircraft position without a space "
            "(example: a0, a1...): " << endl;
        while (true) {  // take a infinite loop for satisfying the valid input for air craft position
            getline(cin, shipPosition);  // get the ship position
            // position length should the 2 character length
            if (shipPosition.length() == POSITION_LENGTH) {
                // make the uppercase of the input position for comparing value
                // and allow for lower or upper case character
                for (auto& c : shipPosition) c = toupper(c);
                if ((shipPosition[0] >= 'A' && shipPosition[0] <= 'J')
                    && (shipPosition[1] >= '0' && shipPosition[1] <= '9')) { // check for valid input
                    break; // if valid input then exit the infinite while loop
                }
                else {
                    cout << "Enter a valid aircraft position without a space"
                        " (example: a0, a1...): \n"; // ask for valid input
                    shipPosition.clear();
                    continue;
                }
            } //end if
            else {
                cout << "Enter a valid aircraft position without a"
                    " space (example: a0, a1...): \n"; // ask for valid input
                shipPosition.clear();
                continue;
            }


        }//end while-loop

        if (tolower(shipOrientiation) == 'h') { // check for horizontal setup
            // Get the start value of y-axis (0,1,2...9)
            //from the letter position (A,B,C....J)
            int startPositionRow = letterToRowNumber(shipPosition[0]);
            int startPositionCol = shipPosition[1] - '0';
            // not able to setup the aircraft horizontally from this position
            if (startPositionCol > AIRCRAFT_LENGTH) {
                cout << "You cannot place the aircraft in this position."
                    " TRY AGAIN!" << endl;
                shipPosition.clear();
                continue;
            } //end if
            else {
                int counter = 0;
                for (int i = startPositionCol;
                    i < startPositionCol + AIRCRAFT_LENGTH; i++) {
                    // set the player matrix with 'A' for indicating the aircraft location
                    playerMatrix[startPositionRow][i] = 'A';
```

```
                // insert the ship position values in the structure variable
                b.aircraft[counter][0] = startPositionRow;
                b.aircraft[counter][1] = i;
                counter++;  // increase the counter one
            }
            break;
        } //end else
    } //end if
    if (tolower(shipOrientiation) == 'v') { // check for the vertical setup
        int startPositionRow = letterToRowNumber(shipPosition[0]);
        int startPositionCol = shipPosition[1] - '0'; // make a character value to integer
        // not able to setup the aircraft vertically from this position
        if (startPositionRow > AIRCRAFT_LENGTH) {
            cout << "You cannot place the aircraft in this position."
                " TRY AGAIN!" << endl;
            shipPosition.clear();
            continue;
        } //end if
        else {
            int counter = 0;
            for (int i = startPositionRow;
                i < startPositionRow + AIRCRAFT_LENGTH; i++) {
                // set the player matrix with 'A' for indicating the aircraft location
                playerMatrix[i][startPositionCol] = 'A';
                b.aircraft[counter][0] = i;// insert the ship position values in member variables
                b.aircraft[counter][1] = startPositionCol;
                counter++; // increase the counter one
            } //end for
            break;
        } //end else

    } //end if
} //end while-loop

objBShip.showPlayZone(playerMatrix); // redraw the player area with the position of battleship
shipOrientiation = '\0';     // reset the ship orientation
shipPosition.clear();        // clear the shipPosition
while (true) {               // loop for setup the battleship position
    cout << "Setup your battleship carrier location" << endl;
    cout << "Select your battleship carrier orientation"
        " (h-horizontal) and (v-vertical) : " << endl;
    while (true) {    // take a infinite loop for satisfying the valid input for ship orientation
        cin >> shipOrientiation;  // take the input of ship orientation 'h' or 'v'
        if (tolower(shipOrientiation) == 'h' ||
            tolower(shipOrientiation) == 'v') {
            cin.ignore();
            break;
        } //end if
        else {
            cout << "Please enter a valid input 'H' or 'h' or 'V' or 'v' \n";
            cin.ignore();
            continue;
        } //end else
    }
```

```
cout << "Enter the battleship position without a space"
    " (example: a0, a1...): " << endl;
while (true) {  // take a infinite loop for satisfying the valid input for battleship position
    getline(cin, shipPosition); // get the ship position
    // position length should the 2 character length
    if (shipPosition.length() == POSITION_LENGTH) {
        for (auto& c : shipPosition) c = toupper(c);
        if ((shipPosition[0] >= 'A' && shipPosition[0] <= 'J')
            && (shipPosition[1] >= '0' && shipPosition[1] <= '9')) { // check for valid input
            break; // if valid input then exit the infinite while loop
        } //end if
        else {
            // ask for valid input again
            cout << "Enter a valid battleship position without a"
                " space (example: a0, a1...): " << endl;
            shipPosition.clear();
            continue;
        } //end else
    }
    else {
        cout << "Enter a valid battleship position without a space"
            " (example: a0, a1...): \n";// ask for valid input again
        shipPosition.clear();
        continue;
    }

} //end while-loop

if (tolower(shipOrientiation) == 'h') {  // check for horizontal setup
// Get the start value of y-axis (0,1,2...9) from the letter position (A,B,C....J)
    int startPositionRow = letterToRowNumber(shipPosition[0]);
    int startPositionCol = shipPosition[1] - '0';
    bool isConflict = true;
    if (startPositionCol + BATTLESHIP_LENGTH <= 10) // check the position withing correct range
    {
        isConflict = conflictWithOtherShip(playerMatrix,
            startPositionRow, startPositionCol, BATTLESHIP_LENGTH,
            tolower(shipOrientiation)); // check the ship is conflict with other ship position
    }
    if (isConflict) // not able to setup the battleship horizontally from this position
    {
        cout << "You cannot place the battleship in this position."
            "TRY AGAIN!" << endl;
        shipPosition.clear();
        continue;
    }
    else
    {
        int counter = 0;
        for (int i = startPositionCol;
            i < startPositionCol + BATTLESHIP_LENGTH; i++)
        {
            // set the player matrix with 'B' for indicating the battle location
            playerMatrix[startPositionRow][i] = 'B';
```

```
                // insert the ship position values in the structure variable
                b.battleship[counter][0] = startPositionRow;
                b.battleship[counter][1] = i;
                counter++; // increase the counter one
            } //end for
            break;
        } //end else
    } //end if
    if (tolower(shipOrientiation) == 'v')   // check for the vertical setup
    {
        int startPositionRow = letterToRowNumber(shipPosition[0]);
        int startPositionCol = shipPosition[1] - '0'; // make a character value to integer
        bool isConflict = true;
        if (startPositionRow + BATTLESHIP_LENGTH <= 10)// check the position withing correct range
        {
            isConflict = conflictWithOtherShip(playerMatrix,
                startPositionRow, startPositionCol, BATTLESHIP_LENGTH,
                tolower(shipOrientiation)); // check the ship is conflict with other ship position
        }
        if (isConflict) // not able to setup the battleship vertically from this position
        {
            cout << "You cannot place the battleship in this position."
                " TRY AGAIN!" << endl;
            shipPosition.clear();
            continue;
        }
        else {
            int counter = 0;
            for (int i = startPositionRow; i < startPositionRow
                + BATTLESHIP_LENGTH; i++) {
                // set the player matrix with 'B' for indicating the battle location
                playerMatrix[i][startPositionCol] = 'B';
                b.battleship[counter][0] = i;
                b.battleship[counter][1] = startPositionCol;
                counter++;   // increase the counter one
            } //end  for
            break;
        } //end else

    } //end if
} //end while
objBShip.showPlayZone(playerMatrix);  // redraw the player area with the position of destroyer
shipOrientiation = '\0';      // reset the ship orientation
shipPosition.clear();   // clear the shipPosition
while (true)   // loop for setup the battleship position
{
    cout << "Setup your destroyer carrier location" << endl;
    cout << "Select your destroyer carrier orientation (h-horizontal)"
        " and (v-vertical) : " << endl;

    while (true) { // take a infinite loop for satisfying the valid input for ship orientation
        cin >> shipOrientiation; // take the input of ship orientation 'h' or 'v'
        if (tolower(shipOrientiation) == 'h' ||
            tolower(shipOrientiation) == 'v') {
```

```
        cin.ignore();
        break;
    }
    else {
        cout << "Please enter a valid input 'H' or 'h' or 'V' or 'v' \n";
        cin.ignore();
        continue;
    }
}
cout << "Enter the destroyer position without a space "
    "(example: a0, a1...): " << endl;
while (true) { // take a infinite loop for satisfying the valid input for destroyer position
    getline(cin, shipPosition);   // get the ship position
    if (shipPosition.length() == POSITION_LENGTH) {
        for (auto& c : shipPosition) c = toupper(c);
        if ((shipPosition[0] >= 'A' && shipPosition[0] <= 'J')
            && (shipPosition[1] >= '0' && shipPosition[1] <= '9')) { // check for valid input
            break; // if valid input then exit the infinite while loop
        }
        else {
            cout << "Enter a valid destroyer position without a space"
                " (example: a0, a1...): \n"; // ask for valid input again
            shipPosition.clear();
            continue;
        }
    }
    else {
        cout << "Enter a valid destroyer position without a space"
            " (example: a0, a1...): " << endl;   // ask for valid input again
        shipPosition.clear();
        continue;
    } //endl else

}

if (tolower(shipOrientiation) == 'h') { // check for horizontal setup
    int startPositionRow = letterToRowNumber(shipPosition[0]);
    int startPositionCol = shipPosition[1] - '0';
    bool isConflict = true;
    if (startPositionCol + DESTROYER_LENGTH <= 10)
    {
        isConflict = conflictWithOtherShip(playerMatrix,
            startPositionRow, startPositionCol, DESTROYER_LENGTH,
            tolower(shipOrientiation)); // check the ship is conflict with other ship position
    }
    if (isConflict) { // not able to setup the battleship horizontally from this position
        cout << "You cannot place the battleship in this position."
            " TRY AGAIN!" << endl;
        shipPosition.clear();
        continue;
    } //end if
    else
    {
        int counter = 0;
```

```
            for (int i = startPositionCol; i < startPositionCol
                + DESTROYER_LENGTH; i++) {
                // set the player matrix with 'D' for indicating the battle location
                playerMatrix[startPositionRow][i] = 'D';
                // insert the ship position values in the member variables
                b.destroyer[counter][0] = startPositionRow;
                b.destroyer[counter][1] = i;
                counter++;  // increase the counter one
            }
            break;
        } //end else
    }
    if (tolower(shipOrientiation) == 'v') // check for the vertical setup
    {
        int startPositionRow = letterToRowNumber(shipPosition[0]);
        int startPositionCol = shipPosition[1] - '0';  // make a character value to integer
        bool isConflict = true;
        if (startPositionRow + DESTROYER_LENGTH <= 10)
        {
            isConflict = conflictWithOtherShip(playerMatrix,
                startPositionRow, startPositionCol, DESTROYER_LENGTH,
                tolower(shipOrientiation)); // check the ship is conflict with other ship position
        }
        if (isConflict) { // not able to setup the destroyer vertically from this position

            cout << "You cannot place the battleship in this position."
                " TRY AGAIN!" << endl;
            shipPosition.clear();
            continue;
        }
        else
        {
            int counter = 0;
            for (int i = startPositionRow; i < startPositionRow
                + DESTROYER_LENGTH; i++) {
                // set the player matrix with 'D' for indicating the battle location
                playerMatrix[i][startPositionCol] = 'D';
                // insert the ship position values in the structure variable
                b.destroyer[counter][0] = i;
                b.destroyer[counter][1] = startPositionCol;
                counter++;// increase the counter one
            }
            break;
        }

    }
}

objBShip.showPlayZone(playerMatrix); // redraw the player area with the position of CORVETTE
shipOrientiation = '\0';     // reset the ship orientation
shipPosition.clear();   // clear the shipPosition
while (true)         // loop for setup the battleship position
{
    cout << "Setup your corvette carrier location" << endl;
cout << "Select your corvette carrier orientation "
    "(h-horizontal) and (v-vertical) : " << endl;
while (true) { // take a infinite loop for satisfying the valid input for ship orientation
    cin >> shipOrientiation; // take the input of ship orientation 'h' or 'v'
    if (tolower(shipOrientiation) == 'h' ||
        tolower(shipOrientiation) == 'v') {
        cin.ignore();
        break;
    }
    else {
        cout << "Please enter a valid input 'H' or 'h' or 'V' or 'v' \n";
        cin.ignore();
        continue;
    }
}
cout << "Enter the corvette position without a space "
    "(example: a0, a1...): " << endl;
while (true) {  // take a infinite loop for satisfying the valid input for corvette position
    getline(cin, shipPosition);      // get the ship position
    // position length should the 2 character length
    if (shipPosition.length() == POSITION_LENGTH) {
        for (auto& c : shipPosition) c = toupper(c);
        if ((shipPosition[0] >= 'A' && shipPosition[0] <= 'J')
            && (shipPosition[1] >= '0' && shipPosition[1] <= '9')) { // check for valid input
            break;      // if valid input then exit the infinite while loop
        }
        else {
            cout << "Enter a valid corvette position without a space "
                "(example: a0, a1...): " << endl;   // ask for valid input again
            shipPosition.clear();
            continue;
        }
    }
    else {
        cout << "Enter a valid corvette position without a space"
            " (example: a0, a1...): " << endl;   // ask for valid input again
        shipPosition.clear();
        continue;
    }

}

if (tolower(shipOrientiation) == 'h')  // check for horizontal setup
{
    int startPositionRow = letterToRowNumber(shipPosition[0]);
    int startPositionCol = shipPosition[1] - '0';
    bool isConflict = true;
    if (startPositionCol + CORVETTE_LENGTH <= 10)
    {
        isConflict = conflictWithOtherShip(playerMatrix,
            startPositionRow, startPositionCol, CORVETTE_LENGTH,
            tolower(shipOrientiation)); // check the ship is conflict with other ship position
    }
    if (isConflict) {  // not able to setup the battleship horizontally from this position
```

```
            cout << "You cannot place the corvette in this position. TRY AGAIN!" << endl;
            shipPosition.clear();
            continue;
        }
        else
        {
            int counter = 0;
            for (int i = startPositionCol; i < startPositionCol
                + CORVETTE_LENGTH; i++) {
                // set the player matrix with 'C' for indicating the battle location
                playerMatrix[startPositionRow][i] = 'C';
                // insert the ship position values in the member variable
                b.corvette[counter][0] = startPositionRow;
                b.corvette[counter][1] = i;
                counter++; // increase the counter one
            }
            break;
        }
    }
    if (tolower(shipOrientiation) == 'v') { // check for the vertical setup
        int startPositionRow = letterToRowNumber(shipPosition[0]);
        int startPositionCol = shipPosition[1] - '0'; // make a character value to integer
        bool isConflict = true;
        if (startPositionRow + CORVETTE_LENGTH <= 10) {
            isConflict = conflictWithOtherShip(playerMatrix,
                startPositionRow, startPositionCol, CORVETTE_LENGTH,
                tolower(shipOrientiation)); // check the ship is conflict with other ship position
        }
        if (isConflict) { // not able to setup the destroyer vertically from this position
            cout << "You cannot place the battleship in this position."
                " TRY AGAIN!" << endl;
            shipPosition.clear();
            continue;
        }
        else
        {
            int counter = 0;
            for (int i = startPositionRow; i < startPositionRow +
                CORVETTE_LENGTH; i++) {
                // set the player matrix with 'C' for indicating the battle location
                playerMatrix[i][startPositionCol] = 'C';
                // insert the ship position values in the member variable
                b.corvette[counter][0] = i;
                b.corvette[counter][1] = startPositionCol;
                counter++;  // increase the counter one
            }
            break;
        }

    }
} //end while-loop
objBShip.showPlayZone(playerMatrix); // redraw the player area with the position of destroyer

return b;
```

```
} //end of setYourBattleShip

//***************************************************************
//Definition of function setComputerBattleShip                    *
//Set computer ship position and return ship position             *
//***************************************************************
BShipSetUp BShipSetUp::setComputerBattleShip(BShipSetUp computerShipPosition) {
    const int AIRCRAFT_LENGTH = 5;      // Unit length of the aircraft
    const int BATTLESHIP_LENGTH = 4;    // Unit length of the battleship
    const int DESTROYER_LENGTH = 3;    // Unit length of the destroyer
    const int CORVETTE_LENGTH = 2;     // Unit length of the corvette
    computerMatrix = getMatrixData();

    //setup aircraft
    int rowPosition = rand() % 2 + 2; // Randomly select a row position from 2-3
    int colPosition = rand() % 2 + 2; // Randomly select a column position from 2-3
    int shipOrientation = rand() % 2; // Select a ship orientation value 0 or 1
    if (shipOrientation == 0) // If value is 0 then consider the orientation as horizontal;
    {
        int counter = 0;
        for (int i = colPosition; i < colPosition + AIRCRAFT_LENGTH; i++) {
            // Set the computer matrix with 'A' for indicating the aircraft location
            computerMatrix[rowPosition][i] = 'A';
            // Insert the ship position values in the structure variable
            computerShipPosition.aircraft[counter][0] = rowPosition;
            computerShipPosition.aircraft[counter][1] = i;  // Insert the column position
            counter++;              // Increase the counter one
        }
    }
    else { // Otherwise orientation is vertical
        int counter = 0;
        for (int i = rowPosition; i < rowPosition + AIRCRAFT_LENGTH; i++)
        {
            // Set the player matrix with 'A' for indicating the battle location
            computerMatrix[i][colPosition] = 'A';
            computerShipPosition.aircraft[counter][0] = i;
            computerShipPosition.aircraft[counter][1] = colPosition;
            counter++;
        }
    }

    //setup the battleship
    rowPosition = (rand() % 2) + 5; // Randomly select a row position from 5-6
    colPosition = (rand() % 2) + 5; // Randomly select a column position from 5-6
    shipOrientation = (rand() % 2); // Select a ship orientation value 0 or 1
    if (shipOrientation == 0) { // If value is 0 then consider the orientation as horizontal;
        int counter = 0;
        for (int i = colPosition; i < colPosition + BATTLESHIP_LENGTH; i++) {
            // Set the computer matrix with 'B' for indicating the battleship location
            computerMatrix[rowPosition][i] = 'B';
            computerShipPosition.battleship[counter][0] = rowPosition;
            computerShipPosition.battleship[counter][1] = i; // Insert the column position
            counter++;          // Increase the counter one
        }
```

```
}
else {  // Otherwise orientation is vertical
    int counter = 0;
    for (int i = rowPosition; i < rowPosition + BATTLESHIP_LENGTH; i++)
    {
        computerMatrix[i][colPosition] = 'B';
        computerShipPosition.battleship[counter][0] = i;
        computerShipPosition.battleship[counter][1] = colPosition;
        counter++;
    }
}

//setup the corvette
rowPosition = (rand() % 2) + 8;  // Randomly select a row position from 8-9
colPosition = (rand() % 3);     // Randomly select a column position from 0-2
shipOrientation = (rand() % 2);
if (shipOrientation == 0)  // If value is 0 then consider the orientation as horizontal;
{
    int counter = 0;
    for (int i = colPosition; i < colPosition + DESTROYER_LENGTH; i++)
    {
        computerMatrix[rowPosition][i] = 'D';
        computerShipPosition.destroyer[counter][0] = rowPosition;
        computerShipPosition.destroyer[counter][1] = i;  // Insert the column position
        counter++;     // Increase the counter one
    }
}
else { // Otherwise orientation is vertical
    rowPosition = (rand() % 3); // Randomly select a row position from 0-2
    colPosition = (rand() % 2) + 8; // Randomly select a column position from 8-9
    int counter = 0;
    for (int i = rowPosition; i < rowPosition + DESTROYER_LENGTH; i++) {
        // Set the player matrix with 'D' for indicating the battle location
        computerMatrix[i][colPosition] = 'D';
        // Insert the ship position values in member variables
        computerShipPosition.destroyer[counter][0] = i;
        computerShipPosition.destroyer[counter][1] = colPosition;
        counter++; // Increase the counter one
    }
}
//setup the destroyer
shipOrientation = (rand() % 2); // Randomly select ship orientation for destroyer
if (shipOrientation == 0) // If value is 0 then consider the orientation as horizontal;
{
    rowPosition = (rand() % 2); // Randomly select a row position from 0-1
    colPosition = (rand() % 7);  // Randomly select a column position from 0-6
    int counter = 0;
    for (int i = colPosition; i < colPosition + CORVETTE_LENGTH; i++) {
        // Set the computer matrix with 'C' for indicating the battleship location
        computerMatrix[rowPosition][i] = 'C';
        // Insert the ship position values in class members variable
        computerShipPosition.corvette[counter][0] = rowPosition;
        computerShipPosition.corvette[counter][1] = i;  // Insert the column position
        counter++;          // Increase the counter one
```

```
            }
        }
        else { // Otherwise orientation is vertical
            rowPosition = (rand() % 5) + 2; // Randomly select a row position from 2-6
            colPosition = (rand() % 2);  // Randomly select a column position from 0-1
            int counter = 0;
            for (int i = rowPosition; i < rowPosition + CORVETTE_LENGTH; i++)
            {
                computerMatrix[i][colPosition] = 'C';
                computerShipPosition.corvette[counter][0] = i;
                computerShipPosition.corvette[counter][1] = colPosition;
                counter++;              // Increase the counter one
            }
        }
        return computerShipPosition;
}

//******************************************************************
//Definition of function showPlayZone, member function of BShipSetUp
//This draw the computer and player play zone
//******************************************************************
void BShipSetUp::showPlayZone() {
     cout << "~~'*'=unexplore area ~~ 'o'=unsuccessful attack ~~ "
        "'@'=successful attack ~~\n";
    cout <<"------ PRESS 'S' (UPPER CASE) TO SAVE THE GAME ANY TIME --------\n";
    cout <<"..............................................................\n";
    cout << " -       Computer Zone                         "
        "  Your Zone       -" << endl;
    cout << " - - - 0 1 2 3 4 5 6 7 8 9 -          "
        " - - - 0 1 2 3 4 5 6 7 8 9 -" << endl;
    for (int i = 0; i < ROWS; i++) {  //Nested loop draw the player and computer matrix
        cout << " | " << yCoord[i] << " | "; // Set the y-column 'A' to 'J'
        for (int j = 0; j < COLS; j++) {
            // Hide the computer ship position
            if (computerMatrix[i][j] >= 'A' && computerMatrix[i][j] <= 'D')
            {
                cout << "* ";   // Display the computer matrix
            }
            else {
                cout << computerMatrix[i][j] << " "; // Display the computer matrix
            }
        }
        cout << "|           | " << yCoord[i] << " | "; //set the y-column 'A' to 'J'
        for (int j = 0; j < COLS; j++)
            cout << playerMatrix[i][j] << " "; // Display the player matrix
        cout << "|" << endl;
    }

}

//****************************************************************
//Definition of function startPlay. Use two BShipSetUp objects as
//@param which allow to input player and computer to
//attack position each other. Return none
```

```
//******************************************************************
void BShipSetUp::startPlay(BShipSetUp battleShipPlayer, BShipSetUp
    battleShipComputer) {
    const int POSITION_LENGTH = 2; // The input position string length always two,e.g, a0,b9..
    int rowPosition;                 // row value
    int colPosition;                // column value
    string attackPosition;          // input string for attack
    bool isSuccessful;              // Successfully attack or not
    int trackWin;                   // The winner of the game
    fstream gInfo;                  // Declare a fstream object
    int attackCount = 0;            // A counter for count the number of attack
    DateTimeInfo <string> dt;       // Declare a DateTimeInfo class object
    char* winner = new char[2];     // Dynamically allocate character array

    if (!battleShipPlayer.getPlayerName().empty()) { // Check for openning a saved game
        playerName = battleShipPlayer.getPlayerName(); // set the player name
    }
    if (battleShipPlayer == battleShipComputer) { // Confirm the both player battle ships are in same size
        cout << "Play is continuing now! " << endl;
    }
    else {
        cout << "Miss configuration of player and computer battle ship";
    }
    dt.setDateOfPlay(currentDate()); // Hold the current date
    dt.setStartTimePlay(currentTime()); // Hold the start time

    cout << "~~ Now your turn to attack the computer ship position ~~" << endl;
    while (true) {
        while (true) { // Take a infinite loop for satisfying the valid input for attack position
            cout << "Choose a position for attacking the computer ships"
                " (example: a0, a1...'S'(save game) ): " << endl;
            getline(cin, attackPosition);                      // Get the ship position
            if (attackPosition == "S") break;
            // position length should the 2 character length
            if (attackPosition.length() == POSITION_LENGTH) {
                // make the uppercase of the input position for comparing value
                //and allow for lower or upper case character
                for (auto& c : attackPosition) c = toupper(c);
                if ((attackPosition[0] >= 'A' && attackPosition[0] <= 'J')
                    && (attackPosition[1] >= '0' && attackPosition[1] <= '9')) { // check for valid input
                    // Get the start value of y-axis (0,1,2...9) from the letter position (A,B,C....J)
                    rowPosition = letterToRowNumber(attackPosition[0]);
                    colPosition = attackPosition[1] - '0';
                    attackCount += 1; // Count the attack
                    isSuccessful = checkSuccessful(battleShipComputer, 'p',
                        rowPosition, colPosition);
                    // If successfully hit then player will get another change for attack
                    if (isSuccessful) {
                        //Display the ship status of the game, and track availability
                        trackWin = displayShipStatus(battleShipPlayer,
                            battleShipComputer);
                        showPlayZone();// Draw the play zone
                        cout << " You attack successfully !!! " << endl;
                        attackPosition.clear();
                        if (trackWin == 0) break; // Player won the game; do not need continue
                        continue;
                    } //end if
                    else {
                        trackWin = displayShipStatus(battleShipPlayer,
                            battleShipComputer);
                        showPlayZone();// Draw the play zone
                        cout << "You miss the hit. Now computer's turn! \n";
                        break;// End the player attack, now computer will attack
                    } //end else
                } //end if
                else {
                    cout << "--Enter a valid aircraft position without a space"
                        " (example: a0, a1...'S'(save game) )----\n";// ask for valid input again
                    attackPosition.clear();
                    continue;
                } //end else
            }
            else {
                cout << "----Enter a valid aircraft position without a space"
                    " (example: a0, a1...'S'(save game) )----" << endl;// ask for valid input again
                attackPosition.clear();
                continue;
            }//end else
        } //end while-loop
        if (attackPosition == "S")break;
        if (trackWin == 0) { // The player won
            cout << "Congratulation!!! ~~~" << playerName
                << "~~~ You won this game!!!" << endl;
            winner[0] = 'p';     // Hold player win the game
            winner[1] = '\0';
            break;
        } // end if
        while (true) { // Take a infinite loop for satisfying the valid input for attack position
            cout << "Computer is now attacking your ships...: " << endl;
            rowPosition = rand() % 10; // randomly select a row position from 0-9
            colPosition = rand() % 10; // randomly select a column position from 0-9
            // Check if it is successfully hit or not
            isSuccessful = checkSuccessful(battleShipPlayer, 'c', rowPosition, colPosition);
            if (isSuccessful) {  // If successfully hit then player will get another change for attack
                trackWin = displayShipStatus(battleShipPlayer, battleShipComputer);
                showPlayZone();   // Draw the play zone
                cout << "Computer attack the position: " << yCoord[rowPosition]
                    << colPosition << endl;
                cout << "Computer attack your ship successfully !!! " << endl;
                if (trackWin == 1) break; // Computer won the game; do not need continue
                continue;
            }//end if
            else {
                trackWin = displayShipStatus(battleShipPlayer, battleShipComputer);
                showPlayZone(); // Draw the play zone
                cout << "Computer attack the position: " << yCoord[rowPosition]
                    << colPosition << endl;
                cout << "Computer miss the hit. Now your turn! " << endl;
```

```
                break;  // End the computer attack and player will attack now to player battle ship
            } //end else
        }//end while-loop
        if (trackWin == 1) { // The computer won
            cout << "Congratulation!!! ~~~ Computer ~~~ won this game!!!\n";
            winner[0] = 'c';                  // Hold the winner
            winner[1] = '\0';

            break;
        }//end if
    } //End while-loop
    dt.setEndTimeOfPlay(currentTime());  //Hold the end time of play
    if (attackPosition == "S") {          // If save the game
        winner[0] = 's';                // No one win the game, and is saved game
        winner[1] = '\0';

        GameSummary objGSummary(attackCount, winner, &dt);
        int rVal = objGSummary.writeSummary(playerName, gInfo);   // Write a summary of game
        // save the game
        saveGame(battleShipPlayer, battleShipComputer, playerMatrix, computerMatrix);
    } //end if
    else {
        GameSummary objGSummary(attackCount, winner, &dt);
        int rVal = objGSummary.writeSummary(playerName, gInfo); //Play is ended, write a summary
    }

} //End of startPlay member function

//*****************************************************************
//Definition of function checkSuccessful                          *
//This function check the attack is successful or not and         *
//Update the ship position structure                              *
//*****************************************************************
bool BShipSetUp::checkSuccessful(BShipSetUp& bShipInfo,
    char sourceData, int row, int col) {
    const int AIRCRAFT_LENGTH = 5;    // Unit length of the aircraft
    const int BATTLESHIP_LENGTH = 4;  // Unit length of the battleship
    const int DESTROYER_LENGTH = 3;   // Unit length of the destroyer
    const int CORVETTE_LENGTH = 2;   // Unit length of the corvette
    bool isFound = false;
    for (int i = 0; i < AIRCRAFT_LENGTH; i++) { // Check the air craft position
        if (bShipInfo.aircraft[i][0] != -1)     // If the position not hit yet
        {
            //hit successful
            if (bShipInfo.aircraft[i][0] == row && bShipInfo.aircraft[i][1] == col)
            {
                // Track this position destroy
                bShipInfo.aircraft[i][0] = -1;
                if (sourceData == 'c') {
                    // Set the matrix position '@' if hit successfully
                    playerMatrix[row][col] = '@';
                }
                else
                {
```

```
                // Set the matrix position '@' if hit successfully
                computerMatrix[row][col] = '@';
            }
            isFound = true;  // The value is found already
            return isFound;
        }
    }
}

for (int i = 0; i < BATTLESHIP_LENGTH; i++) // Check the battle ship position
{
    if (bShipInfo.battleship[i][0] != -1)   // If the position not hit yet
    {
        if (bShipInfo.battleship[i][0] == row &&
            bShipInfo.battleship[i][1] == col) { //hit successful

            bShipInfo.battleship[i][0] = -1;   // Track this position destroy
            if (sourceData == 'c')
            {
                playerMatrix[row][col] = '@'; // Set the matrix position '@' if hit successfully
            }
            else
            {
                computerMatrix[row][col] = '@'; // Set the matrix position '@' if hit successfully
            }
            isFound = true;  // The value is found already
            return isFound;
        }
    }
}

for (int i = 0; i < DESTROYER_LENGTH; i++)     // Check the Destroyer position
{
    if (bShipInfo.destroyer[i][0] != -1)   // If position not hit yet
    {
        if (bShipInfo.destroyer[i][0] == row && bShipInfo.destroyer[i][1] == col) //hit successful
        {
            bShipInfo.destroyer[i][0] = -1;   // Track this position destroy
            if (sourceData == 'c')
            {
                playerMatrix[row][col] = '@';  // Set the matrix position '@' if hit successfully
            }
            else
            {
                computerMatrix[row][col] = '@'; // Set the matrix position '@' if hit successfully
            }
            isFound = true;    // The value is found already
            return isFound;
        }
    }
}

for (int i = 0; i < CORVETTE_LENGTH; i++)     // Check the corvette position
{
        if (bShipInfo.corvette[i][0] != -1)   // If position not hit yet
        {
            if (bShipInfo.corvette[i][0] == row && bShipInfo.corvette[i][1] == col) //hit successful
            {
                bShipInfo.corvette[i][0] = -1;   // Track this position destroy
                if (sourceData == 'c')
                {
                    playerMatrix[row][col] = '@'; // Set the matrix position '@' if hit successfully
                }
                else {
                    computerMatrix[row][col] = '@';  // Set the matrix position '@' if hit successfully
                }
                isFound = true; // The value is found already
                return isFound;
            }
        }
    }
    if (sourceData == 'c')
    {
        playerMatrix[row][col] = 'o'; // Set the matrix position '@' if hit successfully
    }
    else
    {
        computerMatrix[row][col] = 'o'; // Set the matrix position '@' if hit successfully
    }
    return isFound;
}
//*****************************************************************
//Definition of function displayShipStatus
//Check the ship status using structure reference
//Draw the ship status
//Return the integer indicating the game is over or not
//*****************************************************************
int BShipSetUp::displayShipStatus(BShipSetUp player, BShipSetUp computer)
{
    system("cls");
    const int AIRCRAFT_LENGTH = 5;           // Unit length of the aircraft
    const int BATTLESHIP_LENGTH = 4;         // Unit length of the battleship
    const int DESTROYER_LENGTH = 3;          // Unit length of the destroyer
    const int CORVETTE_LENGTH = 2;           // Unit length of the corvette
    int airCarftComputer = 0;                // Counter for air craft
    int battelShipComputer = 0;              // Counter for battleship
    int destroyerComputer = 0;               // Counter for destroyer
    int corvetteComputer = 0;                // Counter for corvette
    int airCarftPlayer = 0;                  // Counter for air craft
    int battelShipPlayer = 0;                // Counter for battleship
    int destroyerPlayer = 0;                 // Counter for destroyer
    int corvettePlayer = 0;                  // Counter for corvette

    bool isFound = false;

    //Count aircraft
    for (int i = 0; i < AIRCRAFT_LENGTH; i++)   // Check the air craft position
    {
```

```
    if (computer.aircraft[i][0] != -1)       // If the position not hit yet
    {
        airCarftComputer += 1;               // Count the computer air craft
    }
    if (player.aircraft[i][0] != -1)
    {
        airCarftPlayer += 1;                 // Count the player air craft
    }
}
//Count battleship
for (int i = 0; i < BATTLESHIP_LENGTH; i++)  // Check the battle ship position
{
    if (computer.battleship[i][0] != -1)      // If the position not hit yet
    {
        battelShipComputer += 1;              // Count the computer battle ship
    }
    if (player.battleship[i][0] != -1)
    {
        battelShipPlayer += 1;               // Count the player battle ship
    }
}
//Count destroyer
for (int i = 0; i < DESTROYER_LENGTH; i++) // Check the Destroyer position
{
    if (computer.destroyer[i][0] != -1)     // If the position not hit yet
    {
        destroyerComputer += 1;            // Count the computer destroyer ship
    }
    if (player.destroyer[i][0] != -1)
    {
        destroyerPlayer += 1;            // Count the player destroyer ship
    }
}
//Count corvette
for (int i = 0; i < CORVETTE_LENGTH; i++)  // Check the corvette position
{
    if (computer.corvette[i][0] != -1)      // If the position not hit yet
    {
        corvetteComputer += 1;              // Count the computer corvette ship
    }
    if (player.corvette[i][0] != -1)
    {
        corvettePlayer += 1;               // Count the player corvette ship
    }
}
//Display the ship status
cout << "........................................................."
    "..........." << endl;
cout << " -      Computer Ships Status                  Your Ships Status-\n";
cout << "........................................................"
    "............" << endl;
cout << "----Aircraft: " << airCarftComputer << " units                 "
    " | ----Aircraft: " << airCarftPlayer << " units" << endl;
cout << "----Battleship: " << battelShipComputer << " units                "
```

```
        " | ----Battleship: " << battelShipPlayer << " units" << endl;
    cout << "----Destroyer: " << destroyerComputer << " units                  "
        "| ----Destroyer: " << destroyerPlayer << " units" << endl;
    cout << "----Corvette: " << corvetteComputer << " units                  "
        " | ----Corvette: " << corvettePlayer << " units" << endl;
    cout << "..............................................................................."
        "......." << endl;
    cout << " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n";
    cout << " -                       ~~~Welcome to BattleShip~~~                 -\n";
    if (player.getPlayerName() == "") {  // Start with player name using new game
        cout << " -                       ~~~Player Name: " << playerName << endl;
    }
    else { //Start with player name using a saved game
        cout << " -                       ~~~Player Name: "
            << player.getPlayerName() << endl;
    }
    cout << " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n";
    // Check all computer ships were hit successfully
    if (airCarftComputer == 0 && battelShipComputer == 0 &&
        destroyerComputer == 0 && corvetteComputer == 0) {
        return 0;   // Computer loss and player win
    }
    // Check all player ships were hit successfully
    else if (airCarftPlayer == 0 && battelShipPlayer == 0 &&
        destroyerPlayer == 0 && corvettePlayer == 0)
    {
        return 1;   // player loss and computer win
    }
    else
    {
        return 2; // Continue play
    }
}

//****************************************************************************
//Definition of function currentDate
//Input->: None, This function convert the current date as MM-DD-YYYY format
//Output->: Return string sTime
//****************************************************************************
string BShipSetUp::currentDate() {
    string sTime;
    time_t curr_time;
    curr_time = time(NULL);
    tm* tm_local = localtime(&curr_time);
    sTime = to_string(tm_local->tm_mon + 1) + "/" + to_string(tm_local->tm_mday)
        + "/" + to_string(tm_local->tm_year % 100);
    return sTime;
} // End currentDate function

//*******************************************************************
//Definition of function currentTime
//Input: None, This function convert the current time as HH:mm:ss format
//Output: Return string sTime
//*******************************************************************
```

```
string BShipSetUp::currentTime() {
    string sTime;
    time_t curr_time;
    curr_time = time(NULL);
    tm* tm_local = localtime(&curr_time);
    sTime = to_string(tm_local->tm_hour) + ":" + to_string(tm_local->tm_min) +
        ":" + to_string(tm_local->tm_sec);
    return sTime;

} //End currentTime function

//*****************************************************************
//Definition of function saveGame. Use structure variables, double
//pointer as parameters, and write the game in binary file.
//Return none
//*****************************************************************
void BShipSetUp::saveGame(BShipSetUp player, BShipSetUp computer,
    vector<vector<char>> playerMatrix, vector<vector<char>> pcMatrix) {
    //setPMatrix(playerMatrix);
    //setCMatrix(computerMatrix);
    char pMat[10][10];
    char cMat[10][10];
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            pMat[i][j] = playerMatrix[i][j];
            cMat[i][j] = pcMatrix[i][j];
        }//end for-loop
    } //end for-loop

    fstream pShip, pMatrix, cShip, cMatrix; //Declare fstream objects to write out
    //open player ship file
    pShip.open("playerShip.txt", ios::out);
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 2; j++)
            pShip << player.aircraft[i][j] << ";"; // Store aircraft position from save game
    pShip << endl;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 2; j++)
            pShip << player.battleship[i][j] << ";"; // Store battleship position from the save game
    pShip << endl;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 2; j++)
            pShip << player.destroyer[i][j] << ";";// Store destroyer postion from the save game
    pShip << endl;
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            pShip << player.corvette[i][j] << ";"; // Store corvette position from the save game
    pShip << endl;
    // Open the computer ship file
    cShip.open("computerShip.txt", ios::out);
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 2; j++)
            cShip << computer.aircraft[i][j] << ";";
    cShip << endl;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 2; j++)
            cShip << computer.battleship[i][j] << ";";
    cShip << endl;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 2; j++)
            cShip << computer.destroyer[i][j] << ";";
    cShip << endl;
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            cShip << computer.corvette[i][j] << ";";

    pMatrix.open("playerMatrix.dat", ios::out | ios::binary);
    cMatrix.open("pcMatrix.dat", ios::out | ios::binary);
    pMatrix.write(reinterpret_cast<char*>(&pMat), (ROWS * COLS) * sizeof(char));
    cMatrix.write(reinterpret_cast<char*>(&cMat), (ROWS * COLS) * sizeof(char));
    //close the file
    pShip.close();
    cShip.close();
    pMatrix.close();
    cMatrix.close();
} //End saveGame function
//****************************************************************
//Definition of function openSummaryFile. Use fstream object
//and string parameters.This is friend function of BShipSetUp class
//check file open in or not. Return Boolean type status
//****************************************************************
bool openSummaryFile(fstream& infile, string fileName) {
    infile.open(fileName, ios::in);
    if (infile.fail()) return false;
    else               return true;

} //End openSummaryFile function

//********************************************************************************
//Definition of function readContent. This is a friend function of BShipSetUp
//Input->:fstream object as @param, function find the last player name from file
//Output->:Return token as player name
//********************************************************************************
string readContent(fstream& infile) {
    string line;             //To read line from file
    string delimiter = ";"; //To read line until this delimiter
    string token;            //To hold player name
    infile >> line;          //Read line from file
    //Find name and store to the token
    token = line.substr(0, line.find(delimiter));
    return token;   //Return the player name
    infile.close(); //Close the file
} // End readContent function
//******************************************************************
//Definition of function openGame.
//Input->:None, and read the game from binary file.
//Output->:Return Boolean status
//******************************************************************
```

```
bool BShipSetUp::openGame() {
    char plMatrix[10][10]; //To read in from binary file to 2d array
    char coMatrix[10][10]; //To read in from binary file to 2d array
    playerMatrix.resize(10, std::vector<char>(10, '*'));    // allocate memory for the vector
    computerMatrix.resize(10, std::vector<char>(10, '*'));    // allocate memory for the vector

    BShipSetUp objPlayer;   //Declare player object
    BShipSetUp objComputer; // Declare computer object
    fstream pMatrix, cMatrix, pShip, cShip; //Declare fstream object to read in
    string playerName; //Name of player

    //open multiple binary files to read
    pShip.open("playerShip.txt", ios::in);
    cShip.open("computerShip.txt", ios::in);
    pMatrix.open("playerMatrix.dat", ios::in | ios::binary);
    cMatrix.open("pcMatrix.dat", ios::in | ios::binary);
    if (pMatrix.fail() || cMatrix.fail() || pShip.fail() || cShip.fail())
        return false;
    string line;    // Store file content at the line
    int lineCount = 1;  // count the line number
    while (getline(pShip, line))
    {
        // Fill the battle ship status
        if (lineCount == 1)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 5; i++) {
                for (int j = 0; j < 2; j++) {
                    objPlayer.aircraft[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else if (lineCount == 2)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 2; j++) {
                    objPlayer.battleship[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else if (lineCount == 3)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 2; j++) {
                    objPlayer.destroyer[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 2; i++) {
                for (int j = 0; j < 2; j++) {
                    objPlayer.corvette[i][j] = tokens[index];
                    index++;
                }
            }
        }
        lineCount++;
    }
    lineCount = 1;
    while (getline(cShip, line))
    {
        if (lineCount == 1)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 5; i++) {
                for (int j = 0; j < 2; j++) {
                    objComputer.aircraft[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else if (lineCount == 2)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 2; j++) {
                    objComputer.battleship[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else if (lineCount == 3)
        {
            int index = 0;
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 2; j++) {
                    objComputer.destroyer[i][j] = tokens[index];
                    index++;
                }
            }
        }
        else
        {
            int index = 0;
```

```
            vector<int> tokens = split(line, ';');
            for (int i = 0; i < 2; i++) {
                for (int j = 0; j < 2; j++) {
                    objComputer.corvette[i][j] = tokens[index];
                    index++;
                }
            }
        }
        lineCount++;
    }

    //Read in from binary files
    pMatrix.read(reinterpret_cast<char*>(&plMatrix), (ROWS * COLS) * sizeof(char));
    cMatrix.read(reinterpret_cast<char*>(&coMatrix), (ROWS * COLS) * sizeof(char));
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            playerMatrix[i][j] = plMatrix[i][j];
            computerMatrix[i][j] = coMatrix[i][j];
        }
    }
    //close the files
    pMatrix.close();
    cShip.close();
    pMatrix.close();
    cMatrix.close();
    // Send file fstream object as a function reference parameter
    bool fOpen = openSummaryFile(pMatrix, "gameSummary.txt");
    if (fOpen) {
        string pName = readContent(pMatrix);
        objPlayer.setName(pName);
        objComputer.setName(pName);
        //Display the ship status
        int retInit = displayShipStatus(objPlayer, objComputer);
        //Display the player information and game head
        cin.ignore();
        //Display the player and computer matrix
        showPlayZone();
        // Start the play allow player and computer to play
        startPlay(objPlayer, objComputer);
    } //end if
    else {
        cout << "Save game is not available now!";
    } //end else
    return true;
} //End openGame function

//**********************************************************************
//Definition of function openSummaryFile. This function Split a line
//Use a reference string and a char as parameters.Return vector object
//**********************************************************************
vector<int> BShipSetUp::split(const string &str, char sep) {
    vector<int> tokens;
    int i;
    stringstream ss(str);
```

```
        while (ss >> i) {
            tokens.push_back(i);
            if (ss.peek() == sep) {
                ss.ignore();
            }
        }

        return tokens;
}
```

GameSummary.cpp

```
/*
 * File:   GameSummary.cpp
 * Author: Khadiza Akter
 * Created on December 5, 2022, 7:11 PM
 * Purpose: Implementation file of GameSummary class
 */
#include <fstream>  //File I/O
#include <iomanip>  //Format the output
using namespace std;

#include "GameSummary.h"

//******************************************************************
//Definition of function writeSummary. Use structure pointer, fstream
//object as @param. This function write game summary in a file
//Return integer value
//****************************************************************
int GameSummary::writeSummary(string pName, fstream& gmInfo) {
    gmInfo.open("gameSummary.txt", ios::out);
    gmInfo << pName << ";";
    gmInfo << dt->getDateOfPlay()<< ";";
    gmInfo << dt->getStartTimePlay() << ";";
    gmInfo << dt->getEndTimeOfPlay() << ";";
    gmInfo << totalNumberOfAttack << ";";
    gmInfo << winner <<";";
    gmInfo.close();
    return 0;
} // End writeSummary function
```

LastGameSummary.cpp

```
/*
 * File:   LastGameSummary.cpp
 * Author: Khadiza Akter
 * Created on December 5, 2022, 7:19 PM
 * Purpose: Implementation file of LastGameSummary class
 */
#include <string>
#include <sstream>
#include <fstream>
```

```
#include <iostream>
using namespace std;

#include "LastGameSummary.h"

//**********************************************************************
//Definition of operator<< overloading function.Use an ostream reference
//object and a constant LastGameSummary class reference object. this
//return a reference to an ostream object.
//**********************************************************************
ostream& operator<<(ostream& output, const LastGameSummary& objLastSumm) {
    output << "\t\t Player name: " << objLastSumm.plName << endl
        << "\t\t Date: " << objLastSumm.dateOfPlay << endl
        << "\t\t Start time: " << objLastSumm.startTime << endl
        << "\t\t End time: " << objLastSumm.endTime << endl
        << "\t\t Total attack: " << objLastSumm.totalAttack << endl;;
    return output;
}

//****************************************************************
//Definition of function lastWinner.
//Input->:None, and read text file to get the name of winner
//Output->:Return string winner
//****************************************************************
string LastGameSummary::lastWinner() {
    char ch; //to get a char from text file
    string winner;
    string sInfo[6];        // To hold info from file
    string line;            // To hold line from file
    string delimiter = ";"; // For split the line
    int i = 0;              // To track the sInfo array index
    size_t pos = 0;         // To hold the size of sub-string

    fstream file("gameSummary.txt", ios::in); //open file to read in
    if (file.fail()) {//Check file open in or not
        cout << "No information available now" << endl;
        return "";
    }
    //Read from file to string
    getline(file, line);
    // Split the line
    while ((pos = line.find(delimiter)) != std::string::npos) {
        sInfo[i] = line.substr(0, pos);
        line.erase(0, pos + delimiter.length());
        i++;
    }//end while-loop
    if (sInfo[5] == "p") {
        winner = "Human";
    }
    else if (sInfo[5] == "c") {
        winner = "Computer";
    }
    else {
        winner = "None(saved game)";
    }
    file.close();
    return winner;
} //End lastWinner function

//*************************************************************************
//Definition of function lastGameSummary.
//Input->:None, this function write and read text file for last game summary
//Output->:Return none
//*************************************************************************
void LastGameSummary::lastGameSummary() {
    //Declare variable
    string delimiter = ";"; // For split the line
    string name = "";       // To hold name of the player
    char yn;                // ask to get input for updating or not
    char ch;                // To get a char from file
    string sInfo[6];        // To hold info from file
    string line;            // To hold line from file
    int i = 0;              // To track the sInfo array index
    size_t pos = 0;         // To hold the size of sub-string
    //Open file to write and read in simultaneously
    fstream file("gameSummary.txt", ios::in | ios::out);
    if (file.fail()) {
        cout << "No summary available now" << endl;
        return;
    }//end if
    //Read from file to string
    getline(file, line);
    // Split the line
    while ((pos = line.find(delimiter)) != std::string::npos) {
        sInfo[i] = line.substr(0, pos);
        line.erase(0, pos + delimiter.length());
        i++;
    }//end while-loop
    //Display the output to the screen
    LastGameSummary objLastGmSummary(*sInfo, *(sInfo + 1), *(sInfo + 2), *(sInfo + 3), stoi(*(sInfo + 4)));
    LastGameSummary objLastGmSummary_new(objLastGmSummary); // call the copy constructor
    cout << objLastGmSummary_new;
    file.close();//close the file
} //End LastGameSummary function
```

AbsDateTime.h

```
/*
 * File:   AbsDateTime.h
 * Author: Khadiza Akter
 * Created on December 5, 2022, 6:46 PM
 * Purpose: Abstract DateTime Class Specification
 */
#ifndef ABSDATETIME_H
#define ABSDATETIME_H
#include <string>
using namespace std;
```

```
    //#include "BShipSetUp"

    class AbsDateTime {
        public:
            virtual string currentTime()=0;
            virtual string currentDate()=0;

    };

#endif /* ABSDATETIME_H */
```

BattleShip.h

```
    /*
     * File:   BattleShip.h
     * Author: Khadiza Akter
     * Created on December 05, 2022, 6:40 PM
     * Purpose: Specification file of BattleShip base class
     */

    #ifndef BATTLESHIP_H
    #define BATTLESHIP_H
    #include <vector>
    using namespace std;

    class BattleShip {
        protected:
            static char yCoord[11]; // declare a character array for maintain y-coordinate as a character
            vector<vector<char>> arrMatrix; //vector of the player and computer matrix
        public:
            const int ROWS = 10;// Constant rows for 10x10 matrix
            const int COLS = 10;// Constant column for 10x10 matrix
            //constructor fill the vector matrix grid (10x10) with '*' character
            //for computer or player matrix.
            BattleShip() {
                arrMatrix.resize(10, vector<char>(10, '*'));
            }
            // accessor inline function, return filled array matrix
            vector<vector<char>> getMatrixData() const { return arrMatrix; };
            void showPlayZone(vector<vector<char>>); //Display player zone matrix

    };

#endif /* BATTLESHIP_H */
```

BShipSetUp.h

```
    /*
     * File:   BShipSetUp.h
     * Author: Khadiza Akter
     * Created on December 5, 2022, 6:50 PM
     * Purpose: Specification file of derived BShipSetUp class from both BattleShip
              and AbsDateTime class
     */
```

```
#ifndef BSHIPSETUP_H
#define BSHIPSETUP_H
#include <iostream>       //Input-output library
#include <string>         //To work with strings
using namespace std; //Standard Name-space under which System Libraries reside

#include "BattleShip.h"  //needed for base class
#include "AbsDateTime.h" //needed for base class
#include "GameSummary.h" //needed for GameSummary class

//Derived class from both base class BattleShip and AbsDateTime
class BShipSetUp :public BattleShip, public AbsDateTime {// Declare a class for different types battle ship
    private:
        string playerName; // Declare the input array to take player name
        int aircraft[5][2]; //Aircraft length is 5 for tracking its coordinate value
        int battleship[4][2]; // Battleship length is 4 for tracking its coordinate values (row,col)
        int destroyer[3][2]; // Destroyer length is 3 for its coordinate values (row,col)
        int corvette[2][2]; // Corvette length is 2 for tracking its coordinate values (row,col)
        vector<vector<char>> playerMatrix;// to hold player matrix information
        vector<vector<char>> computerMatrix;// to hold computer Matrix information

    public:
        //Exception class for InvalidName
        class InvalidName {};
        //constructor #1
        BShipSetUp() { }
        //constructor #2, check for the validation of player name.if invalid
        //input throw exception, otherwise set name to the playerName
        BShipSetUp(string name) {
            bool space = false;
            if (name.empty()) throw InvalidName();
            else if (name.size() > 0) {
                for (int i = 0; i < name.size(); i++) {
                    if (isspace(name[i])) space = true;
                    else {
                        space = false;
                        break;
                    }
                }
                if (space == true) throw InvalidName();
                else               playerName = name;
            }
        }
        string getPlayerName() const { return playerName; }//accessor function
        // mutator function,and set player ship position and return ship members
        BShipSetUp setYourBattleShip(BShipSetUp);
        //Set computer ship position and return ship members
        BShipSetUp setComputerBattleShip(BShipSetUp);
        //Determine the letter (A,...J) value to integer y-axis value (0,1,2...9)
        friend int letterToRowNumber(char);
         //operator= overloaded
        const BShipSetUp operator=(const BShipSetUp& right);
        bool operator ==(const BShipSetUp& equal);// Operator overloading ==
        // Draw the computer and player zone and same function name
        // as base class that will provide run-time polymorphism
        void showPlayZone();
        // mutator inline function,and set player name
        void setName(string name) { playerName = name; }
        // Allow the player and computer for attack the ships
        void startPlay(BShipSetUp, BShipSetUp);
        bool checkSuccessful(BShipSetUp&, char, int, int); // The attack was successful or not
        int displayShipStatus(BShipSetUp, BShipSetUp); // Display the ship status for computer and player
        void saveGame(BShipSetUp, BShipSetUp, vector<vector<char>>, vector<vector<char>>); // Save the game
        bool openGame(); // Open a saved game
        //Check the ship position conflict with other ship or not
        bool conflictWithOtherShip(vector<vector<char>>, int, int, int, char);
        string currentTime(); // Redefined the virtual method from abstract class
        string currentDate(); // Redefined the virtual method from abstract class
        vector<int> split(const string& str, char sep); //Returned vector from a string using a separator character
        friend bool openSummaryFile(fstream& infile, string fileName); //check file open in or not.
        friend string readContent(fstream& infile); //find the last player name from file
};

#endif /* BSHIPSETUP_H */
```

DateTimeInfo.h

```
/*
 * File:   DateTimeInfo.h
 * Author: Khadiza Akter
 * Created on December 5, 2022, 7:08 PM
 * Purpose: Specification of DateTimeInfo template class
 */

#ifndef DATETIMEINFO_H
#define DATETIMEINFO_H
//#include<string>
//using namespace std;

template <class T>
class DateTimeInfo {  // Declare a class for date time information of game
private:
   T dateOfPlay;      // Playing date
   T startTimeOfPlay; // Starting time of play
   T endTimeOfPlay;   // Ending time of play
public:
   //Mutator functions
   void setDateOfPlay(T dPlay) { dateOfPlay = dPlay; }
   void setStartTimePlay(T sPlay) { startTimeOfPlay = sPlay; }
   void setEndTimeOfPlay(T ePlay) { endTimeOfPlay = ePlay; }
   //Accessor functions
   T getDateOfPlay() const { return dateOfPlay; } // Return the date of play
   T getStartTimePlay() const { return startTimeOfPlay; } // Return the date of play
   T getEndTimeOfPlay() const { return endTimeOfPlay; } // Return the date of play

};
```

```
#endif /* DATETIMEINFO_H */

        /*
         * File:   GameSummary.h
         * Author: Khadiza Akter
         * Created on December 5, 2022, 7:11 PM
         * Purpose: Specification file of GameSummary class
         */

        #ifndef GAMESUMMARY_H
        #define GAMESUMMARY_H
        #include <cstring> //For strlen(), strcpy()
        using namespace std;

        #include "DateTimeInfo.h" //Needed for DateTimeInfo class

        class GameSummary { // Declare a class for game summary
          private:
            DateTimeInfo<string>* dt; // Aggregation; use object reference of DateTimeInfo class
            int totalNumberOfAttack;  // Total number of attack needed for this game
            char *winner;             // Who is the winner of the game
          public:
            //default constructor #1
            GameSummary() {
              totalNumberOfAttack = 0;
            }
            //constructor #2
            GameSummary(int numOfAttck, char* win, DateTimeInfo<string>* dTime) {
              winner = new char[strlen(win) + 1];
              this->totalNumberOfAttack = numOfAttck;
              strcpy(this->winner,win);
              this->dt = dTime;
            }
            int writeSummary(string, fstream&);
            // Destructor for delete winner
            ~GameSummary() {
              //free the memory
              delete winner;
            }
        };

#endif /* GAMESUMMARY_H */
```

LastGameSummary.h

```
        /*
         * File:   LastGameSummary.h
         * Author: Khadiza Akter
         * Created on December 5, 2022, 7:19 PM
         * Purpose: Specification file of LastGameSummary class
         */
```

```
#ifndef LASTGAMESUMMARY_H
#define LASTGAMESUMMARY_H
#include <string>
using namespace std;

class LastGameSummary {
    private:
        string plName;      //Last game's player name
        string dateOfPlay;  // Date of the play
        string startTime;   // Start time of the play
        string endTime;     // End time of the play
        int totalAttack;    // Total number of attack of play
    public:
        LastGameSummary() {}
        //Constructor of the class
        LastGameSummary(string pName, string dPlay, string sTime, string eTime, int tAttack) {
            plName = pName;
            dateOfPlay = dPlay;
            startTime = sTime;
            endTime = eTime;
            totalAttack = tAttack;
        }
        // Copy constructor
        LastGameSummary(LastGameSummary& objLastSumm) {
            plName = objLastSumm.plName;
            dateOfPlay = objLastSumm.dateOfPlay;
            startTime = objLastSumm.startTime;
            endTime = objLastSumm.endTime;
            totalAttack = objLastSumm.totalAttack;
        }
        void lastGameSummary();
        //Friend, Operator<< overloading for displaying data
        friend ostream& operator<<(ostream&, const LastGameSummary&);
        string lastWinner(); // Last winner of the game

};

#endif /* LASTGAMESUMMARY_H */
```