AQA WARSHIPS

Description of the Program

The program is designed to play a game which is similar to Battleships.

There are five ships hidden on a 10-by-10 board. The players takes shots at different coordinates specified by a column (0—9) and a row (0—9).

The ships and sizes are as follows:

- Aircraft Carrier — 5 cells
- Battleship — 4 cells
- Submarine — 3 cells
- Destroyer — 3 cells
- Patrol Boat — 2 cells

Ships can be either horizontal or vertical on the board.

The program consists of one constant (TRAININGGAME) which holds the filename to be loaded with the data for the board. This is then populated into Board (a two-dimensional array of Chars). The possible values for each cell are: — (empty sea), A (a piece of aircraft carrier), B (a piece of battleship), S (a piece of submarine), D (a piece of destroyer), P (a piece of Patrol Boat), m (an empty square that has already been fired at) and h (a square which contained a piece of ship and has been hit).

The program has two possible starts: the first is where the position of the ships is loaded from a file, and the second where random positions for the ships are generated by the computer. This second method uses a lot of additional code as the ships cannot overlap or go off the board and this is checked by the program.

The game proceeds by asking the player for a column and then a row. The program then checks what is stored at this index in the Board array. If it is a — this is then replaced by an m. If it is a piece of ship (A, B, D, S, P) then this is replaced by an h. If this position already contains an m or an h, a message saying that the user has already fired here is displayed.

If a position off the board is entered, the program will stop functioning.

To complete and end the game you must sink all parts of each ship. There is no limit to the number of shots that a player may take. The player can keep firing until they have hit every square.

Description of Program Elements

The program consists of several routines to determine the validity of moves and who has won.
The program elements that are used are described in order below.

Element	Type	Description
Ships	An array/list	Stores the name and size of all the ships (each location in the array has the name and size for one ship)
Board	A two-dimensional array/list of one character strings	Stores the current state of the board
TRAININGGAME	A string constant	Stores the filename of the training file
MenuOption	An integer variable	Used to store what number the user has selected from the menu
Row	An integer variable	Used to store the row on the board
Column	An integer variable	Used to store the column on the board
Orientation	A string variable	Stores direction of a ship: V for vertical, H for horizontal
HorV	An integer variable	Used to randomly generate the orientation of the ship: a zero becomes V for vertical, a one becomes h for horizontal

Description of Program Routines

The program functions Ⓕ and procedures Ⓟ are described below.

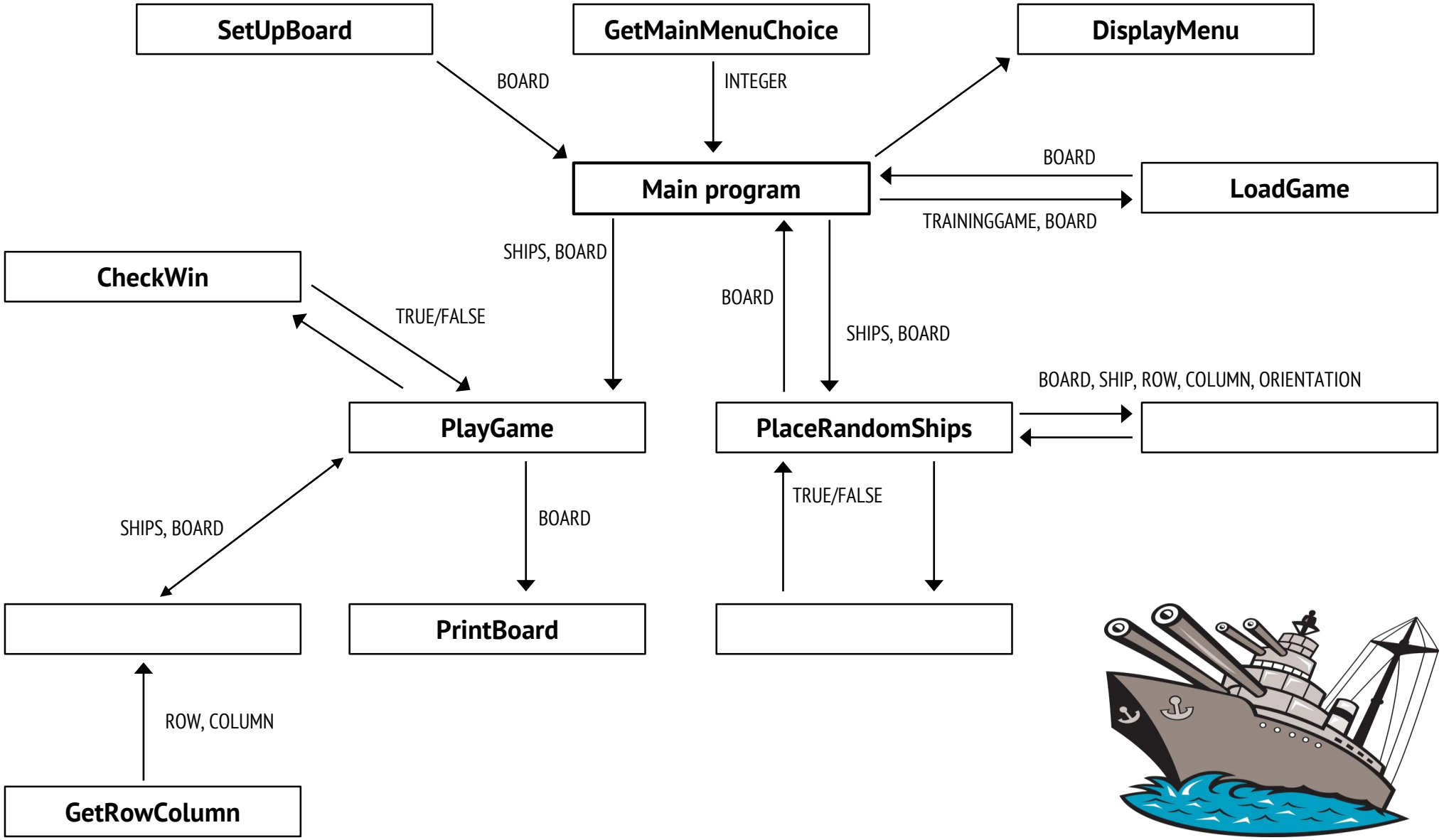
Routine	Description	
CheckWin Ⓕ	Receives: Board Returns: Boolean Called from: PlayGame	Checks every position in Board to see whether any section of ship is left Returns false if it finds a piece Returns true if it checks every position and doesn't find anything
DisplayMenu Ⓟ	Receives: nothing Returns: nothing Called from: main program	A simple procedure that prints options on the screen.

Routine	Description	
GetMainMenuChoice Ⓕ	Receives: nothing Returns: integer Called from: main program	Handles the user's menu choice: 1. Prompts the user to enter an integer 2. Returns that number
GetRowColumn Ⓕ	Receives: nothing Returns: Row, Column Called from: MakePlayerMove	1. Prompts the user for a column and gets a value 2. Prompts the user for a row and gets a value 3. Returns both Row and Column
LoadGame Ⓟ	Receives: Filename, Board Returns: nothing Called from: main program	1. Reads the data contained in the sample file into an object called BoardFile 2. Then chops Line into individual elements and assigns them to the correct location on the board 3. Repeats for all 10 rows 4. Closes the file
MakePlayerMove Ⓟ	Receives: Board, Ships Returns: nothing Called from: PlayGame	1. Receives the row and column from GetRowColumn 2. Checks whether that position has already been fired at 3. Checks whether that position is a miss and changes value stored in board if it is 4. If neither 2 nor 3 are true it must be a hit; changes value stored in board to reflect this
PlaceRandomShips Ⓟ	Receives: Board, Ships Returns: nothing Called from: main program	This procedure is not used when the training game is selected. It generates a random row, column and a third number (HorV) to decide whether the ship runs horizontally or vertically. It then uses the function ValidateBoatPosition to check whether there is already a boat running through that position, and that all of the boat is placed on the board (and doesn't run off the edge). If the position is suitable, the boat is placed using PlaceShip. If not, another position and orientation is generated. This continues until all ships have been placed.
PlaceShip Ⓟ	Receives: Board, Ship, Row, Column, Orientation Returns: nothing Called from: PlaceRandomShips	Places the ships on the board. Uses For loop that counts up to the size of the ship being placed (this is stored in Ship[1]). The loop counter is called Scan. Scan is added to row when placing a vertical ship (so that the column remains the same). Scan is added to column when placing a horizontal ship (so that the row remains the same). The board is populated in occupied positions with the first letter of the name of the ship (stored in Ship[0][0]).

Routine	Description	
PlayGame Ⓟ	Receives: Board, Ships Returns: nothing Called from: main program	Starts a game and keeps it running: 1. Sets the Boolean GameWon to false 2. Starts a conditional loop that keeps checking the value of GameWon and continues while it is false 2.1. Displays the board 2.2. Gets the player to bomb a square (make a move) 2.3. Checks to see whether the game has been won (if it has a success message is displayed and the loop will exit after this iteration)
PrintBoard Ⓟ	Receives: Board Returns: nothing Called from: PlayGame	Displays the board: 1. Starts off by displaying a message saying 'The board looks like this' 2. Next, a For loop is used to print the column headings (0 to 9) 3. Nested For loops now display the board; the first increments the row (top to bottom) 3.1. Prints the row number 3.2. Second For loop works its way along the row 3.2.1. An empty square is displayed as " " 3.2.2. A square with ship in it is also displayed as " " (hiding the location of the ships) 3.2.3. Anything else (a hit (h) or a miss (m)) is then displayed 3.2.4. A separator is displayed unless it is the last column (so no further values to display)
SetUpBoard Ⓕ	Receives: nothing Returns: Board Called from: main program	1. Cycles through all positions on the board using nested For loops 1.1. Assigns all positions on the board to a dash – Some of these dashes will be replaced later when ships are placed on the board.

Routine	Description	
ValidateBoatPosition Ⓕ	Receives: Board, Ship, Row, Column, Orientation Returns: Boolean Called from: PlaceRandomShips	Checks to see whether it is possible to place a boat in that position (Is there another boat in the way? Does the boat run off the edge of the board?) 1. If the row number plus the ship size is greater than 10 and a ship is running vertically then it will go off the edge of the board. The function returns false (not a valid boat position). 2. If the column number plus the ship size is greater than 10 and a ship is running horizontally then it will go off the edge of the board. The function returns false (not a valid boat position). 3. If the ship is vertical: 3.1. A For loop scans along the column from the start of the ship to the end 3.1.1. If a position isn't empty (–) then false is returned (invalid ship position) 4. If the ship is horizontal: 4.1. A For loop scans along the row from the start of the ship to the end 4.1.1. If a position isn't empty (–) then false is returned (invalid ship position) 5. If this part of the function is reached, there is no reason the ship cannot be placed here and true is returned.
Main program Ⓟ	1. Sets up the board 2. Declares a variable to store what menu option has been selected and assigns it to 0 (this could in fact be any number so long as it wasn't 9) 3. Starts a conditional loop that continues until the user selects option 9 (to quit) 3.1. Populates board with data by calling SetUpBoard (this would reset the board too if this was the second game played) 3.2. Displays the menu by calling DisplayMenu 3.3. Calls GetMainMenuChoice to get the user's choice and stores it in the variable menuOption 3.4. If the user picks option 1: 3.4.1. The board is populated by the ships in random locations 3.4.2. The game is started 3.5. If the user picks option 2: 3.5.1. The board is populated from the training text file 3.5.2. The game is started	

AQA WARSHIPS



Programming Theory Questions

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but do not require any additional programming.

1. State the name of an identifier for:

   (a) An array or list variable (1 mark)

   .....

   (b) A subroutine that has five parameters (1 mark)

   .....

   (c) A variable that is used to store a whole number (1 mark)

   .....

   (d) A subroutine that returns one or more values (1 mark)

   .....

   (e) A variable that stores a Boolean value (1 mark)

   .....

2. Look at the function ValidateBoatPosition.

   What is the purpose of the variable Orientation? (2 marks)

   .....

   .....

   .....

3 What data is stored for each ship? (2 marks)

   .....

   .....

   .....

4. Look at the procedure PlayGame.

   What is the purpose of the While loop? (3 marks)

   .....

   .....

   .....

   .....

5. Give an example of a declaration and assignment statement from the Skeleton Program where a variable is assigned an initial value when it is declared. (2 marks)

6. Explain the operation of the procedure PlaceShip. (4 marks)

7. The skeleton program utilises the variable Board.

   (a) Describe the data structure held by Board. (1 mark)

   (b) How is the data stored and used in this structure? (3 marks)

8. State the name of an identifier for:

   (a) A subroutine that contains a nested loop (1 mark)

   (b) A procedure that is passed 2 parameters (1 mark)

   (c) A variable that stores text (1 mark)

   (d) A constant (1 mark)

   (e) A library function with exactly one parameter that returns an integer value (1 mark)

9. Look at the procedure PrintBoard.

(a) What lines of code print the column headings? (3 marks)

.....

.....

.....

(b) What is the advantage of this method over 'hard-coding'? (2 marks)

.....

.....

.....

10. This question is in relation to the routines PlaceRandomShips and LoadGame.

These routines both use a local variable called Row. What are local variables, and in relation to these routines what is an advantage of utilising local variables? (3 marks)

.....

.....

.....

.....

11. The procedure PrintBoard utilises a For loop, whereas the main program utilises a While loop.

What is the difference between a For loop and a While loop? (4 marks)

.....

.....

.....

.....

.....

12. PrintBoard is a procedure, whereas GetMainMenuChoice is a function.

Describe the difference between a procedure and a function. (2 marks)

.....

.....

.....

13. What is the purpose of the following line?

```
BoardFile = open(Filename, "r")
```

(1 mark)

.....

.....

14. What is the purpose of these lines?

```
for Row in range(10):
    Line = BoardFile.readline()
    for Column in range(10):
        Board[Row][Column] = Line[Column]
```

(4 marks)

.....

.....

.....

.....

.....

15. The LoadGame procedure uses the file Training.txt by default.

(a) What would happen to the program if Training.txt did not exist? (1 mark)

.....

.....

(b) Describe how we would change the program to solve this. (3 marks)

.....

.....

.....

.....

TOTAL MARKS

/50

Programming Theory Questions

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but do not require any additional programming.

1. State the name of an identifier for:

   (a) An array or list variable (1 mark)

   (b) A subroutine that has five parameters (1 mark)

   (c) A variable that is used to store a whole number (1 mark)

   (d) A subroutine that returns one or more values (1 mark)

   (e) A variable that stores a Boolean value (1 mark)

2. Look at the function ValidateBoatPosition.

   What is the purpose of the variable Orientation? (2 marks)

3 What data is stored for each ship? (2 marks)

4. Look at the procedure PlayGame.

   What is the purpose of the While loop? (3 marks)

5. Give an example of a declaration and assignment statement from the Skeleton Program where a variable is assigned an initial value when it is declared. (2 marks)

6. Explain the operation of the procedure PlaceShip. (4 marks)

7. The skeleton program utilises the variable Board.

   (a) Describe the data structure held by Board. (1 mark)

   (b) How is the data stored and used in this structure? (3 marks)

8. State the name of an identifier for:

   (a) A subroutine that contains a nested loop (1 mark)

   (b) A procedure that is passed 2 parameters (1 mark)

   (c) A variable that stores text (1 mark)

   (d) A constant (1 mark)

   (e) A library function with exactly one parameter that returns an integer value (1 mark)

9. Look at the procedure PrintBoard.

   (a) What lines of code print the column headings? (3 marks)

   (b) What is the advantage of this method over 'hard-coding'? (2 marks)

10. This question is in relation to the routines PlaceRandomShips and LoadGame.

    These routines both use a local variable called Row. What are local variables, and in relation to these routines what is an advantage of utilising local variables? (3 marks)

11. The procedure PrintBoard utilises a For loop, whereas the main program utilises a While loop.

    What is the difference between a For loop and a While loop? (4 marks)

12. PrintBoard is a procedure, whereas GetMainMenuChoice is a function.

    Describe the difference between a procedure and a function. (2 marks)

13. What is the purpose of the following line?

```
BoardFile = open(Filename, "r")
```

(1 mark)

14. What is the purpose of these lines?

```
for Row in range(10):
    Line = BoardFile.readline()
    for Column in range(10):
      Board[Row][Column] = Line[Column]
```

(4 marks)

15. The LoadGame procedure uses the file Training.txt by default.

    (a) What would happen to the program if Training.txt did not exist? (1 mark)

    (b) Describe how we would change the program to solve this. (3 marks)

TOTAL MARKS

/50

Programming Exercises

The following require you to open the skeleton program and make modifications. They are written in examination style and illustrate how you should prepare your answers.

Question 1

This question refers to GetRowColumn.

It is currently possible to fire at coordinates that are off the board, crashing the game. Amend GetRowColumn so that this is not possible. If a square off the board is targeted, the message: 'Sorry, that is outside the target area. Please select again.' should be displayed and the user prompted to re-enter.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for GetRowColumn
- SCREEN CAPTURE(S) of testing a shot at column 14 row -8

6 marks

Question 2

This question refers to PlayGame.

It is currently possible to fire at every square in order until you find every ship. Alter PlayGame so that the player only has 20 torpedoes. The number of torpedoes should decrease by 1 after every move and be displayed on-screen. When the number of torpedoes reaches 0, the message 'GAME OVER! You ran out of ammo' should be displayed and the game should end.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for PlayGame.
- SCREEN CAPTURE(S) of testing showing the number of torpedoes going down and the 'game over' message

7 marks

Question 3

This question refers to DisplayMenu and the main program.

Alter the menu so that an option 3 is also displayed between options 2 and 9.
The menu should display '3. Load saved game'.
If option 3 is selected, that program should display 'OPTION 3 EXECUTED'.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for DisplayMenu
- SCREEN CAPTURE(S) of testing

4 marks

Question 4

This question refers to the main program.

Alter the procedure so that if the user enters 9 they are prompted with an 'Are you sure?' message. Only if they respond Y will the program quit.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for the main program
- SCREEN CAPTURE(S) of testing

6 marks

Question 5

This question refers to the main program.

Option 3 currently just displays a message. Amend it so that it prompts the user for a filename and then loads this file and plays the game.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for the main program
- SCREEN CAPTURE(S) of testing using the filename 'Training.txt'

5 marks

Question 6

Create a procedure called SaveGame. It should accept the board as a parameter that is passed to it along with a variable called filename.

It should then save the current state of the board to a text file named the value of filename and in the same format as Training.txt.

Evidence you need to provide

- Your SOURCE CODE PROGRAM for SaveGame

8 marks

Question 7

This question refers to PlayGame.

After a player has made a move, they should be prompted: 'Do you want to save the game (Y, N)?'
If the player enters Y, they should then be prompted for a filename and the game saved using the procedure created in Question 6.

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for PlayGame
- SCREEN CAPTURE(S) of loading a game saved by the user

6 marks

Question 8

This question refers to multiple sections of the skeleton code.

Create a menu option '4. Board Test'. It will set up a board and then display the real values stored on a randomly generated board (revealing the location of the ships). After the board has been displayed the program will return to the main menu. A procedure called RealBoard (similar to PrintBoard) should be created to display the board.

Evidence you need to provide

- Your amended sections of SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) of testing

10 marks

Question 9

This question refers to multiple sections of the skeleton code.

A new ship has joined the fleet called a Frigate. It has a length of 3. Amend the program so that a Frigate is placed in addition to the original ships when option 1 or 4 is selected. 'F' will represent a piece of Frigate.

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) using menu option 4 to show the Frigate

5 marks

Question 10

This question refers to MakePlayerMove.

When a player misses, a radar scan of the adjacent cells should be performed. If any contain an undiscovered section of ship, the message 'Enemy Near!' should be displayed. If not, the message 'All quiet' should be displayed. You should create a function called RadarScan that returns a Boolean value for this operation (True = enemy near).

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for MakePlayerMove
- Your new SOURCE CODE PROGRAM for RadarScan
- SCREEN CAPTURE(S) showing both types of radar scan message

13 marks

Question 11

This question refers to MakePlayerMove.

When a ship is hit its type must be displayed, e.g.:
Hit Aircraft Carrier at (8,6)

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) of a successful hit and the message

8 marks

Question 12

This question refers to PlaceShip, ValidateBoatPosition and PlaceRandomShips.

Amend the program so that all ships can be placed diagonally down and to the left. They still cannot go off the board or overlap with other ships, e.g.:

B			
	B		
		B	
			B

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) of a board generated by option 4 showing at least one diagonal ship

14 marks

Question 13

This question refers to MakePlayerMove.

Amend the program so that if a ship is hit its size is reduced by 1.
A message will then display how many pieces of the ship are left to hit.

e.g.
Hit Battleship at (5,3)
There are 3 pieces of Battleship left

When the size reaches zero an additional message should say that the ship has been sunk.

e.g.
Hit Battleship at (5,6)
There are 0 pieces of Battleship left
YOU SANK THE BATTLESHIP

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) of a ship being sunk

12 marks

Question 14

This question refers to multiple sections of the skeleton code.

A new menu option needs to be added: '5. Manually place ships'.

When selected the user will be prompted for the starting square and orientation of each ship in turn. The program will then check whether this location is valid using ValidateBoatPosition. If a suitable location is selected, a message will confirm that the ship is placed and then place the ship using PlaceShip.

e.g. Aircraft Carrier successfully placed at (1,3)

If ValidateBoatPosition returns false an error message will be displayed.
e.g. Invalid location. Please choose again.

After each ship has been placed, the RealBoard procedure should display the position of all placed ships.

When all ships are placed the game should begin.

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) showing the board before and after the submarine is placed

17 marks

Question 15

This question refers to multiple sections of the skeleton code.

Create a variable to store the current player's score. Everybody starts at 0. Add 1 to score for each shot. A lower score is better.

Create a user-defined data structure (similar to ship) called score.
It should contain a name and a score in suitable data types.

An array/list of five scores will store the scores.

Create a procedure (similar to SetUpBoard) called SetUpScores. It should populate the scores with the following data. It should only do this once when the program is first run.

George	17
Paul	19
John	23
Ringo	25
Bryan	35

Create a menu option '6. Display high-score table' that executes a suitable procedure.

Create a procedure to bubble-sort the high-score table called BubSortScores.

If a player scores less than somebody on the table (remember that a lower score is better) then the worst score on the table will be replaced with their name (you will need to prompt for this) and score, and the table sorted using BubSortScores.

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes
- SCREEN CAPTURE(S) showing the table being displayed before and after a new high score is added

34 marks

