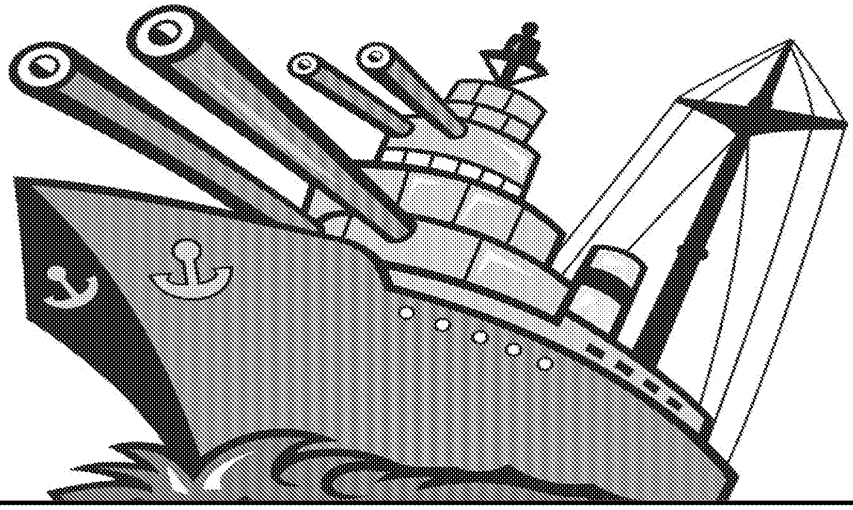
Computer Science

AS | AQA | 7516



2015 specification
for the 2016 AS exam

PAPER 1 EXAM RESOURCE PACK 2016

AQA WARSHIPS

for AS AQA Computer Science

JAVA

AS2/
6500

POD
6500

zigzageducation.co.uk

Contents

Teacher's Introduction

This pack is designed to help you support your students taking the AQA Computing Paper 1 examination.
It is based on the AQA Paper 1 'AQA Warships' preliminary material (JAVA) – for examination June 2016.

① **Pre-release Commentary** (for teachers)
A detailed overview of the skeleton program, describing all JAVA code elements and routines.

This section is designed to help you get to grips with the program, so that you can feel confident helping your students. This commentary is not designed to be given to students before they have explored the code for themselves, and if used in this way could lead to misconceptions of how the program works.

② **Structure Chart Activity**
A partially incomplete diagram for students to complete while getting to grips with the skeleton program. Any missing routines and variables must be added to the diagram. A completed version is provided in the solutions section at the back of the resource.

③ **Programming Theory Questions**
Theory questions test students' understanding of the 'AQA Warships' code, like Section B in the Paper 1 exam. These are provided in both write-on and non-write-on format.

④ **Programming Exercises**
Modification exercises put students' programming skills to the test, like Section C in the Paper 1 exam. An Electronic Answer Document (EAD) and the modified JAVA code are provided on the CD.

Answers and solutions for the structure chart activity, theory questions and programming exercises are provided from page 21 onwards. Note that for the programming exercises in particular, these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

The **Appendices** contains some additional resources, including:

- Further modifications worksheet: a template for brainstorming further enhancements to the skeleton program. This is suggested as a group activity, so that students (and the teacher) can share their ideas, thus increasing the likelihood of covering every area that will come up in the exam.
- Electronic Answer Document (EAD) printout: hard copy version of the file on CD (for reference).

The accompanying CD includes the following files (inside the JAVA folder):

- **MODIFIED_JAVA_CODE.txt** – text file containing the additional and/or modified program code as shown in the mark scheme for section ④ (from page 24).
- **PAPER1_EAD.docx** – Electronic Answer Document for completing sections ③ and ④

This resource is intended to supplement your teaching only. It is the teacher's responsibility to decide how to use this resource to assist themselves and their students appropriately. You may simply wish to read this material to better inform yourself and to help you prepare your lessons and to give you ideas for your teaching. You may also consider whether it is appropriate to hand out some of the sheets for reference and to use some of the activities for classwork or homework. You may also consider whether it is appropriate to hand out the booklet to be worked through by your students more independently. As with all pre-release material, it is the teacher's responsibility to decide in what way to assist their students, and to decide how this resource in particular can be used to fit into that assistance.

The resources here are provided as an interpretation of the pre-release material. The author does not have any special knowledge of what to expect on any particular exam.

Programming Exercises: Teacher

Suggested Question Combinations

It is not envisaged that a student would complete all questions in a 1-hour perio
One approach is to get students to work through all the questions under 'open-b
be followed up by setting combinations of the questions under test conditions si

- No access to previously created code
- No access to notes
- No access to the Internet
- No collaboration
- Strict time limit

Suggested question combinations and time limits for these tests are as follows:

Q1, Q2 & Q4	25 minutes
Q3, Q5, Q6 & Q7	30 minutes
Q8 & Q9	20 minutes
Q10 & Q11	25 minutes

Q8 & Q12	30 minutes
Q13 & Q15	60 minutes
Q8 & Q14	35 minutes

It is also useful (and fun) to get students to come out and solve a question 'live' a
classmates.

Possible Additional Questions

1. When the game has finished, tell the user how accurate they were as a perce
hits by the total number of shots. E.g. 10 hits, 30 shots = 33% hit rate. Only d
2. One shot sinks a ship.
3. Sea mine is placed on the board. If the player hits it, they lose and the game
4. Change the game so the fleet is five Battleships.
5. Create a two-player game.
6. Change the blast radius so that a torpedo also hits ships in adjacent squares
7. Change the dimensions of the board.
8. Create the option to send a sonar ping down a column or row which tempo
ships.
9. Add an ammo store to the board. If the player hits it they get 10 more torpe
10. Change the program so that both coordinates are entered as one input.
11. Make each ship type have a default orientation.
12. Ask for the user's name at the start of the game, and when they win show th
[name]!"
13. Allow user to go back to the main menu
14. Change the torpedo to a missile that obliterates a 9 square block.
15. Change the game so that the user places the ships and the computer fires th
16. Adapt the missile task (above) so that the user can choose whether to use a
fire a maximum of 2 missiles
17. Add a main menu option which will allow you to select which ships are to be
18. Enhance the computer player in task 15 further so that if it hits a square it w
squares until a ship is sunk

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 diffe column (0—9) and a row (0-

The ships and their 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 the board. This is then populated into board (a two-dimensional array of Chars) cell are: — (empty sea), A (a piece of aircraft carrier), B (a piece of battleship), S (of destroyer), P (a piece of Patrol Boat), m (an empty square that has already bee contained a piece of ship and has been hit).

The program has two possible starts: the first is where the position of the ships i second where random positions for the ships are generated by the computer. T additional code as the ships cannot overlap or go off the board and this is check

The game proceeds by asking the player for a column and then a row. The prog at this index in the board array. If it is a — this is then replaced by an m. If it is a this is replaced by an h. If this position already contains an m or an h, a message 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 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
Ship	User-defined data type which contains the data name and size	Stores the name and size of a ship
ships	An array of Ship	Stores the name and size of all the shi
board	A two-dimensional array of characters	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 ha
row	An integer variable	Used to store the row on the board
column	An integer variable	Used to store the column on the board
move	An integer array	Used to store both the row and the col and column must be grouped together
orientation	A char variable	Stores direction of a ship: V for vertical
HorV	An integer variable	Used to randomly generate the orienta horizontal

Description of Program Routines

The program functions Ⓕ and procedures Ⓟ are described below.

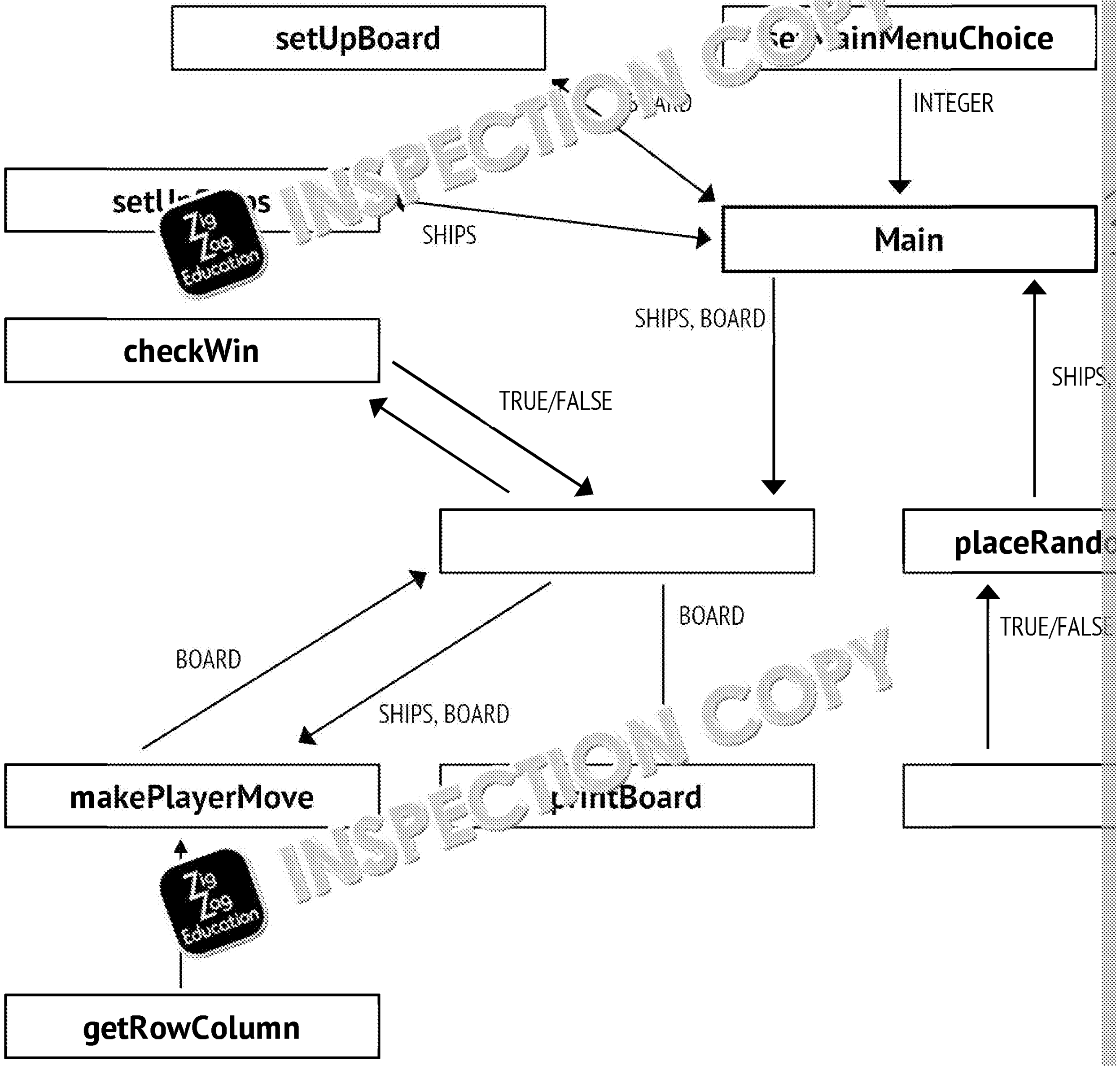
Routine	Description	
checkWin Ⓕ	Receives: board, ships Returns: Boolean Called from: playGame	Checks every position in board to Returns false if it finds a piece Returns true if it checks every pos
displayMenu Ⓟ	Receives: nothing Returns: nothing Called from: main	A simple procedure that prints op
getMainMenuChoice Ⓕ	Receives: nothing Returns: integer Called from: main	Handles the user's menu choice: 1. Prompts the user to ente 2. Returns that number
getRowColumn Ⓕ	Receives: nothing Returns: integer array Called from: makePlayerMove	1. Prompts the user for a co 2. Prompts the user for a ro 3. Returns these two values
loadGame Ⓟ	Receives: filename, board Returns: nothing Called from: main	1. Reads the data contained 2. Reads in a row of data 3. Then chops that line into board 4. Repeats for all 10 rows 5. Closes the file
makePlayerMove Ⓟ	Receives: board, ships Returns: nothing Called from: playGame	1. Receives the row and col 2. Splits them into separate 3. Checks whether that posi 4. Checks whether that posi 5. If neither 3 nor 4 are true

Routine	Description	
placeRandomShips ⓟ	Receives: board, ships Returns: nothing Called from: main	This method is not It generates a rand ship runs horizont It then uses the fu running through th doesn't run off the If not, another pos been placed.
placeShip ⓟ	Receives: board, ship, row, column, orientation Returns: nothing Called from: placeRandomShips	Places the ships o Uses For loop that ship.size). The loo ship (so that the co horizontal ship (so The board is popu ship.
playGame ⓟ	Receives: board, ships Returns: nothing Called from: main	Starts a game and 1. Sets the Boole 2. Starts a While it is false 2.1. Displays 2.2. Gets the 2.3. Checks to GameWo 2.4. Prints a s

Routine	Description	
printBoard Ⓟ	Receives: board Returns: nothing Called from: playGame	Displays the board: 1. Starts off by displaying a mes 2. Next, a For loop is used to pr 3. Nested For loops now display 3.1. Prints the row number 3.2. Second For loop works i 3.2.1. An empty square is 3.2.2. A square with ship 3.2.3. Anything else (a hit 3.2.4. A separator is displ
setUpBoard Ⓟ	Receives: board Returns: nothing Called from: main	1. Cycles through all positions o 1.1. Assigns all positions on Some of these dashes will be repl
setUpShips Ⓟ	Receives: ships Returns: nothing Called from: main	Initialises the ships in the array (u Sets the name of each ship Sets the size of each ship
validateBoatPosition Ⓕ	Receives: board, ship, row, column, orientation Returns: Boolean Called from: placeRandomShips	Checks to see whether it is possib Does the boat run off the edge of 1. If the row number plus the sh goes off the edge of the board. 2. If the column number plus th it will go off the edge of the b 3. If the ship is vertical: 3.1. A For loop scans along t 3.1.1. If a position isn't er 4. If the ship is horizontal: 4.1. A For loop scans along t 4.1.1. If a position isn't er 5. If this part of the function is r returned.

Routine	Description
Main	1. Sets the constant TrainingGame to the correct filename; the default 2. Declares (creates) an empty two-dimensional array of chars to store 3. Declares (creates) an empty array of ships to store the fleet details 4. Declares a variable to store which menu option has been selected an 9) 5. Starts a While loop that continues until the user selects option 9 (to 5.1. Populates board with data by calling setUpBoard (this would re 5.2. Populates ships with data by calling setUpShips 5.3. Displays the menu by calling displayMenu 5.4. Calls getMenuChoice to get the user's choice and stores it in the 5.5. If the user picks option 1: 5.5.1. The board is populated by the ships in random locations 5.5.2. The game is started 5.6. If the user picks option 2: 5.6.1. The board is populated from the training text file 5.6.2. The game is started

AQA WARSH


setUpBoard
ge ainMenuChoice
INTEGER
ARD
setU ps
SHIPS
Main
SHIPS, BOARD
checkWin
TRUE/FALSE
SHIPS
placeRand
BOARD
TRUE/FALS
BOARD
SHIPS, BOARD
makePlayerMove
intBoard
getRowColumn

Programming Theory Questio

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

1. State the name of an identifier for:

   (a) An array or list variable

   ....

   (b) A subroutine that has five parameters

   ....

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

   ....

   (d) A subroutine that returns one or more values

   ....

   (e) A variable that stores a Boolean value

   ....

2. Look at the method validateBoatPosition.

   What is the purpose of the variable orientation?

   ....

   ....

   ....

3 What data is stored for each ship?

   ....

   ....

   ....

4. Look at the method playGame

   What is the purpose of the While loop?

   ....

   ....

   ....

   ....

5. Give an example of a declaration and assignment statement from the Skele
variable is assigned an initial value when it is declared.

....

....

....

6. Explain the operation of the method placeShip.

....

....

....

....

....

7. The skeleton program utilises the variable board.

   (a) Describe the data structure held by board.

   ....

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

   ....

   ....

   ....

   ....

8. State the name of an identifier for:

   (a) A subroutine that contains a nested loop

   ....

   (b) A user-defined data type

   ....

   (c) A variable that stores text

   ....

   (d) A constant

   ....

   (e) A library function with exactly one parameter that returns an integer v

   ....

9. Look at the method printBoard.

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

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

10. This question is in relation to the routines placeRandomShips and loadGar
    These routines both use a local variable called row. What are local variables,
    to these routines what is an advantage of utilising local variables?

11. The procedure printBoard utilises a For loop, whereas the Main procedure u
    What is the difference between a For loop and a While loop?

12. setUpShips is a procedure, whereas getMainMenuChoice is a function.
    Describe the difference between a procedure and a function.

13. What is the purpose of the following line?

```
boardFile.closeFile();
```

....

....

14. What is the purpose of these lines?

```
String line = boardFile.readLine();
for(int column = 0; column<10; column++){
  board[row][column] = line.charAt(column);
}
```

....

....

....

....

....

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

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

....

....

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

....

....

....

....

Programming Theory Questio

These questions refer to the Preliminary Material and require you to loa
but do not require any additional programming.

1. State the name of an identifier for:

   (a) An array or list variable

   (b) A subroutine that has five parameters

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

   (d) A subroutine that returns one or more values

   (e) A variable that stores a Boolean value

2. Look at the method validateBoatPosition.
   What is the purpose of the variable orientation?

3 What data is stored for each ship?

4. Look at the method playGame.
   What is the purpose of the While loop?

5. Give an example of a declaration and assignment statement from the Skele
   variable is assigned an initial value when it is declared.

6. Explain the operation of the method placeShip.

7. The skeleton program utilises the variable board.

   (a) Describe the data structure held by board.

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

8. State the name of an identifier for:

   (a) A subroutine that contains a nested loop

   (b) A user-defined data type

   (c) A variable that stores text

   (d) A constant

   (e) A library function with exactly one parameter that returns an integer v

9. Look at the method printBoard.

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

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

10. This question is in relation to the routines placeRandomShips and loadGa

These routines both use a local variable called row. What are local variables to these routines what is an advantage of utilising local variables?

11. The procedure printBoard utilises a For loop, whereas the Main procedure u

What is the difference between a For loop and a While loop?

12. setUpShips is a procedure, whereas getMainMenuChoice is a function.

Describe the difference between a procedure and a function.

13. What is the purpose of the following line?

```
boardFile.close();
```

14. What is the purpose of these lines?

```
String line = boardFile.readLine();
for(int column = 0; column<10; column++){
  board[row][column] = line.charAt(column);
}
```

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

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

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

Programming Exercises

The following require you to open the skeleton program and make modifications. Th
and illustrate how you should prepare your answer

Question 1

This question refers to getRowColumn.

It is currently possible to fire at coordinates that are off the board, crashing the g that this is not possible. If a square off the board is targeted, the message: 'Sorry, 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

Question 2

This question refers to playGame.

It is currently possible to fire at every square in order until you find every ship. A only has 20 torpedoes. The number of torpedoes should decrease by 1 after eve screen. When the number of torpedoes reaches 0, the message 'GAME OVER! Yo 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 d message

Question 3

This question refers to displayMenu and Main.

Alter the menu so that 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 and Main
- SCREEN CAPTURE(S) of testing

Question 4

This question refers to Main.

Alter the procedure so that if the user enters 9 they are prompted with an 'Are y
respond Y will the program quit.

Evidence you need to provide

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

Question 5

This question refers to Main.

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

Evidence you need to provide

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

Question 6

Create a procedure called saveGame. It should accept the board as a parameter
variable called filename.

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

You should use AQAWriteTextFile2016.

Evidence you need to provide

- Your SOURCE CODE PROGRAM for saveGame

Question 7

This question refers to playGame.

After a player has made a move, they should be prompted: 'Do you want to save
If the player enters Y, they should then be prompted for a filename and the gam
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

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 generated board (revealing the location of the ships). After the board has been return to the main menu. A procedure called realBoard (similar to printBoard) s board.

Evidence you need to provide

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

Question

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 placed in addition to the original ships when option 1 or 4 is selected. 'F' will re

Evidence you need to provide

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

Question 10

This question refers to makePlayerMove.

When a player misses, a radar scan of the adjacent cells should be performed. If section of ship, the message 'Enemy Near!' should be displayed. If not, the mess displayed. You should create a function called radarScan that returns a Boolean 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

Question 11

This question refers to playGame.

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 you
- SCREEN CAPTURE(S) of a successful hit and the message

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 l 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 you
- SCREEN CAPTURE(S) of a board generated by option 4 showing at least on

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 b

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 you
- SCREEN CAPTURE(S) of a ship being sunk

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
program will then check whether this location is valid using validateBoatPositio
selected, a message will confirm that the ship is placed and then place the ship

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 realD[illegible] procedure should display the po

When all ships are placed the game should begin.

Evidence you need to provide

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

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
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 method (similar to setUpBoard and setUpShips) called setUpScores. It
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 proc

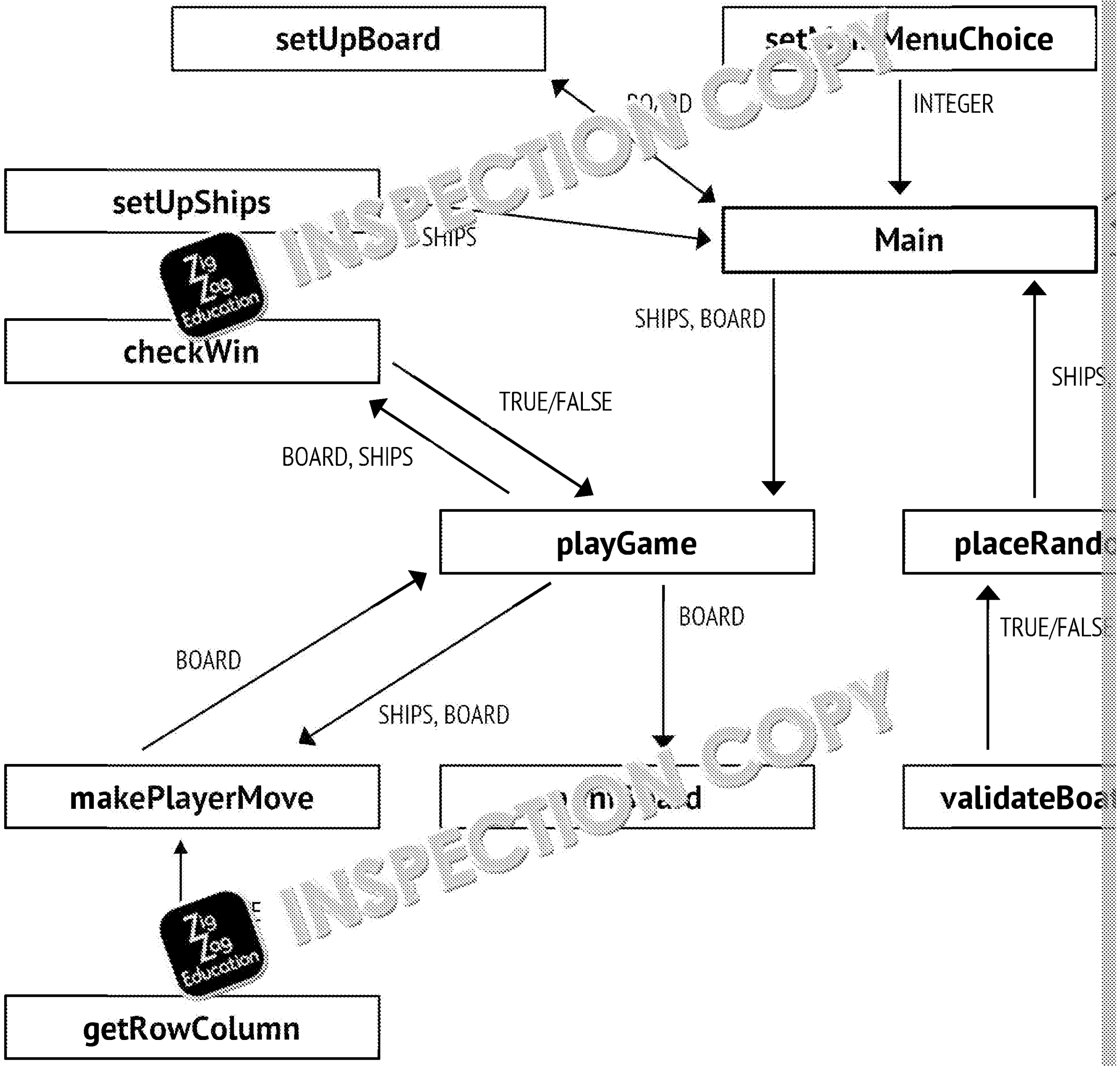
Create a method to bubble sort the high-score table called bubSortScores.

If a player scores less than somebody on the table (remember that a lower score
on the table [illegible] be replaced with their name (you will need to prompt for this)
using bubSortScores.

Evidence you need to provide

- Your amended sections of the SOURCE CODE PROGRAM highlighting you
- SCREEN CAPTURE(S) showing the table being displayed before and after

Structure Chart (Solution)



Programming Theory Questions (Answers)

Q	Marking Guidance
1a	ships / board / move
1b	validateBoatPosition
1c	row / column / HorV / menuOption
1d	getRowColumn / validateBoatPosition / checkWin / getMainMenuChoice
1e	valid / gameWon
2	To store whether the boat should be vertically or horizontally positioned (1 m board (1 mark)
3	Name (1 mark), Size (1 mark)
4	To ensure that the board is printed (1 mark) and the user input requested ag (1 mark) while the game is not yet won (1 mark)

5	Int menuOption = 0;
	Or
	Char orientation = " ";
	Or
	Int row = 0;
	Or
	Int column = 0;

6	To check whether the ship can be placed on the board (1 mark) by ensuring th edge of the board (1 mark) or run across another ship (1 mark). A value of true will only be returned if neither of these situations is the case (1
7a	Character array / char array / 2D array of characters
7b	Any three points (1 mark each): • Two-dimensional array • 10-by-10 array • One dimension for the column • One dimension for the row • A row,column / x,y value is used to refer to each element
8a	loadGame / placeRandomShips
8b	ship (reject Ships; this is an array)
8c	line (reject TrainingGame; this is a constant)
8d	TrainingGame
8e	Random

9a	1 mark for print line, 2 marks for For loop:
	for(int column = 0; column<10; column++)
	{
	console.print(" " + column + " ");
	}

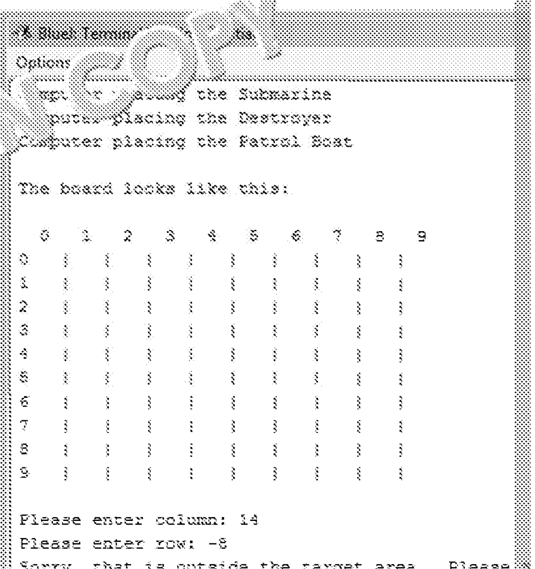
9b	It is easier to modify the game (1 mark), it allows many lines of code to be con (1 mark).

Q	Marking Guidance
10	Local variable: stores a value for only that particular routine. The value is lost (1 mark). Both routines can use the <u>same variable names</u> to traverse the array <u>without a</u> (2 marks for showing understanding of underlined words; 1 mark for partial un
11	A For loop repeats a set number of times (1 mark) and the number of times is before the loop starts (1 mark). A While loop repeats an unknown number of times (1 mark) while a certain co
12	A procedure is a routine called by the program which performs a set of actions A function is a routine called within an expression which returns a result (1 ma
13	The file must be closed after it has been used or it cannot be accessed by othe
14	Reads a line of the training game file (1 mark), then for each column (1 mark) individual characters (1 mark) and assigns them to the correct position on the
15a	It would crash
15b	A try catch (1 mark) should be used to catch the error (1 mark) and then displa (1 mark).

Programming Exercises (Solutions)

Question	Answer
1	***getRowColumn***

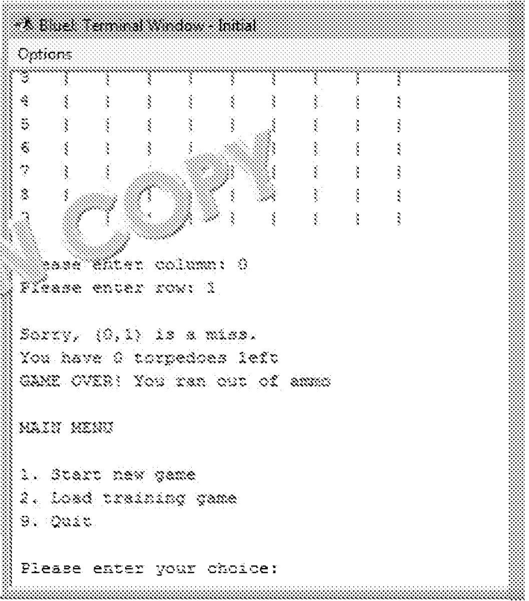
```
int[] getRowColumn(){
   int column;
   int row;
   int[] move;
   move = new int[2];
   console.println();
   column = console.readInteger("Please enter column: ");
   row = console.readInteger("Please enter row: ");
   while ((row > 9) || (row < 0) || (column > 9) || (column < 0))
   {
      console.println("Sorry, that is outside the target area.  Please select again");
      column = console.readInteger("Please enter column: ");
      row = console.readInteger("Please enter row: ");
   }
   console.println();
   move[0] = row;
   move[1] = column;
   return move;
  }
```



```
Computer placing the Submarine
Computer placing the Destroyer
Computer placing the Patrol Boat

The board looks like this:

  0   1   2   3   4   5   6   7   8   9
0   |   |   |   |   |   |   |   |   |   |
1   |   |   |   |   |   |   |   |   |   |
2   |   |   |   |   |   |   |   |   |   |
3   |   |   |   |   |   |   |   |   |   |
4   |   |   |   |   |   |   |   |   |   |
5   |   |   |   |   |   |   |   |   |   |
6   |   |   |   |   |   |   |   |   |   |
7   |   |   |   |   |   |   |   |   |   |
8   |   |   |   |   |   |   |   |   |   |
9   |   |   |   |   |   |   |   |   |   |

Please enter column: 14
Please enter row: -8
Sorry, that is outside the target area.  Please
```

Question	Answer
2	*playGame*

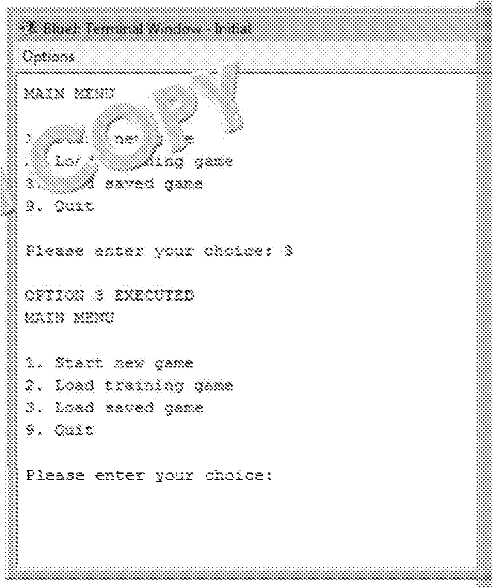
```
void playGame(char[][] board, Ship[] ships){
  boolean GameWon = false;
  int torpedoes = 20;
  while(!GameWon && torpedoes > 0){
    printBoard(board);
    makePlayerMove(board, ships);
    torpedoes = torpedoes - 1;
    console.println("You have " + torpedoes + " torpedoes left");
    GameWon = CheckWin(board, ships);
    if (GameWon){
      console.println("All ships sunk!");
      console.println();
    }
    if (torpedoes == 0)
    {
      console.println("GAME OVER! You ran out of ammo");
      console.println();
    }
  }
}
```



```
3   |   |   |   |   |   |   |   |   |
4   |   |   |   |   |   |   |   |   |
5   |   |   |   |   |   |   |   |   |
6   |   |   |   |   |   |   |   |   |
7   |   |   |   |   |   |   |   |   |
8   |   |   |   |   |   |   |   |   |
9   |   |   |   |   |   |   |   |   |

Please enter column: 0
Please enter row: 1

Sorry, (0,1) is a miss.
You have 0 torpedoes left
GAME OVER! You ran out of ammo

MAIN MENU

1. Start new game
2. Load training game
9. Quit

Please enter your choice:
```

Question	Answer
3	***displayMenu***

```
 void displayMenu(){
  console.println("MAIN MENU");
  console.println();
  console.println("1. Start new game");
  console.println("2. Load training gam[illegible]
  console.println("3. Load sav[illegible]
  console.println("9. Q[illegible]
  console.prir[illegible]
 [illegible]

[illegible]
   if (menuOption == 1){
    placeRandomShips(board, ships);
    playGame(board, ships);
   }
   if (menuOption == 2){
    loadGame(TrainingGame, board);
    playGame(board, ships);
   }
   if (menuOption == 3){
    console.println("OPTION 3 EXECUTED");
   }
  }
```

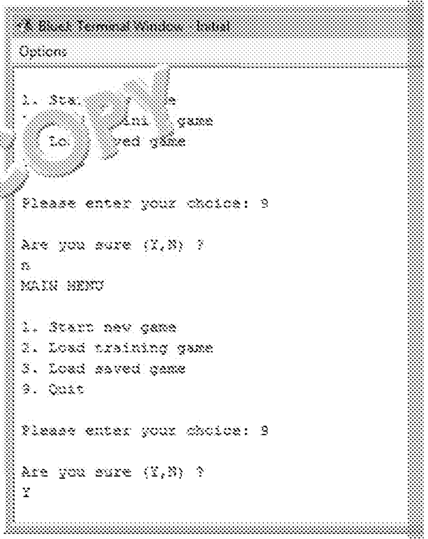


```
MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice: 3

OPTION 3 EXECUTED
MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice:
```

<table>
<tr><th>Question</th><th>Answer</th></tr>
<tr><td>4</td><td>

Main

```
if (menuOption == 9){
      console.println("Are you sure (Y,N) ?");
      char sure = console.readChar();
      if (sure != 'Y')
      {
         menuOption = 0;
      }
   }
```

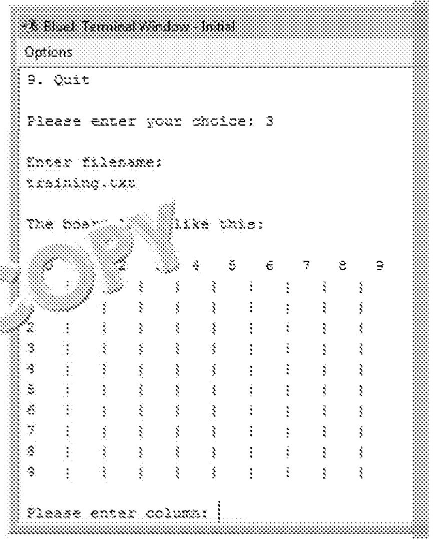


```
1. Start new game
2. Load training game
3. Load saved game

Please enter your choice: 9

Are you sure (Y,N) ?
n
MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice: 9

Are you sure (Y,N) ?
Y
```

</td></tr>
<tr><td>5</td><td>

Main

```
if (menuOption == 3){
      console.println("Enter filename:");
      String fileName = console.readLine();
      loadGame(fileName, board);
      playGame(board, ships);
   }
```



```
9. Quit

Please enter your choice: 3

Enter filename:
training.txt

The board looks like this:

     0   1   2   3   4   5   6   7   8   9
0  |   |   |   |   |   |   |   |   |   |
1  |   |   |   |   |   |   |   |   |   |
2  |   |   |   |   |   |   |   |   |   |
3  |   |   |   |   |   |   |   |   |   |
4  |   |   |   |   |   |   |   |   |   |
5  |   |   |   |   |   |   |   |   |   |
6  |   |   |   |   |   |   |   |   |   |
7  |   |   |   |   |   |   |   |   |   |
8  |   |   |   |   |   |   |   |   |   |
9  |   |   |   |   |   |   |   |   |   |

Please enter column: 
```

</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th></tr>
<tr><td>6</td><td>

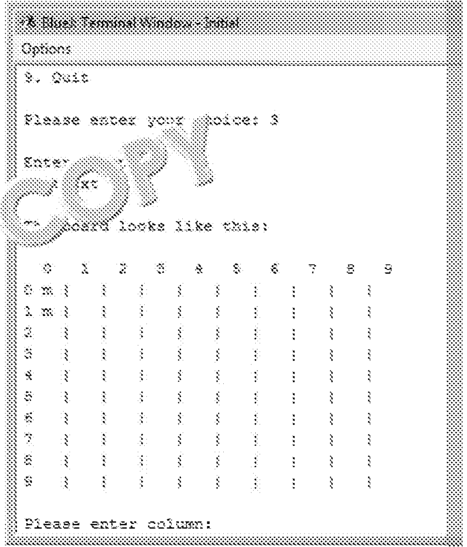
saveGame

```
void saveGame(String filename, char[][] board){
   AQAWriteTextFile2016 boardFile = new AQAWriteTextFile2016(filename);
   for(int row = 0; row<10; row++){
     String line = "";
     for(int column = 0; column<10; column++){
       line = line + board[row][column];
     }
     boardFile.writeToTextFile(line);
   }
   boardFile.closeFile();
```

</td></tr>
<tr><td>7</td><td>

playGame

```
console.println("Do you want to save the game (Y, N)?");
     char save = console.readChar();
     if (save == 'Y')
     {
        console.println("Please enter file name: ");
        String fileName = console.readLine();
        saveGame(fileName, board);
     }
```



```
9. Quit

Please enter your choice: 3

Enter [illegible]
[illegible]xt

The board looks like this:

  0   1   2   3   4   5   6   7   8   9
0 m |   |   |   |   |   |   |   |   |   |
1 m |   |   |   |   |   |   |   |   |   |
2   |   |   |   |   |   |   |   |   |   |
3   |   |   |   |   |   |   |   |   |   |
4   |   |   |   |   |   |   |   |   |   |
5   |   |   |   |   |   |   |   |   |   |
6   |   |   |   |   |   |   |   |   |   |
7   |   |   |   |   |   |   |   |   |   |
8   |   |   |   |   |   |   |   |   |   |
9   |   |   |   |   |   |   |   |   |   |

Please enter column:
```

</td></tr>
</table>

Question	Answer
8	

displayMenu

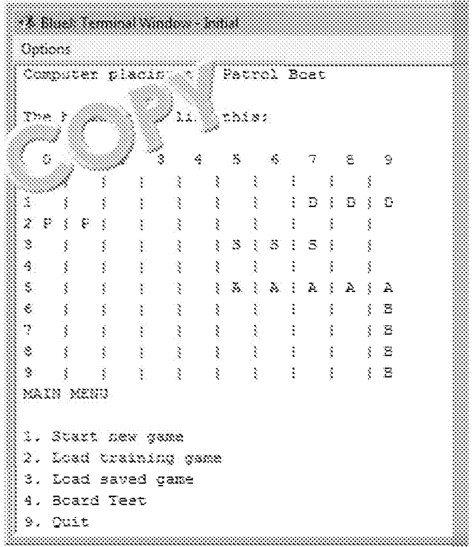
```
console.println("4. Board Test");
```

Main

```
if (menuOption == 4){
        placeRandomShips(board, ships);
        realBoard(board);
    }
```

realBoard

```
  static void realBoard(char[][] board){
    console.println();
    console.println("The board looks like this: ");
    console.println();
    console.print (" ");
    for(int column = 0; column<10; column++){
      console.print(" " + column + "  ");
    }
    console.println();
    for(int row = 0; row<10; row++){
      console.print (row + " ");
      for(int column = 0; column<10; column++){
        if(board[row][column] == '-'){
          console.print(" ");
        }
        //else if (board[row][column] == 'A' || board[row][column] == 'B' ||
        board[row][column] == 'S' || board[row][column] == 'D' || board[row][column] == 'P'){
        //  console.print(" ");
        //}
        else{
          console.print(board[row][column]);
        }
        if(column != 9){
          console.print(" | ");
        }
      }
      console.println();
    }
  }
```



```
BlueJ: Terminal Window - Initial
Options
Computer placing the Patrol Boat


The board looks like this:

  0   1   2   3   4   5   6   7   8   9
    |   |   |   |   |   |   |   |   |
1   |   |   |   |   |   |   | D | D | D
2 P | P |   |   |   |   |   |   |   |
3   |   |   |   |   | S | S | S |   |
4   |   |   |   |   |   |   |   |   |
5   |   |   |   |   | A | A | A | A | A
6   |   |   |   |   |   |   |   |   | B
7   |   |   |   |   |   |   |   |   | B
8   |   |   |   |   |   |   |   |   | B
9   |   |   |   |   |   |   |   |   | B
MAIN MENU


1. Start new game
2. Load training game
3. Load saved game
4. Board Test
9. Quit
```

Question	Answer
9	***setUpShips***

```
ships[5] = new Ship();
ships[5].name = "Frigate";
ships[5].size = 3;
```

Main

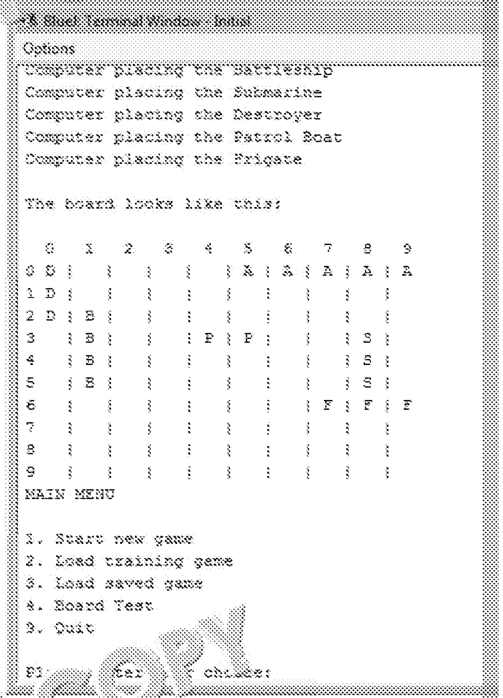
```
Ship ships[] = new Ship[6];
```

checkWin

```
boolean checkWin(char[][] board, Ship[] ships){
  for(int row = 0; row<10; row++){
    for(int column = 0; column<10; column++){
      if(board[row][column] == 'A' || board[row][column] == 'B' || board[row][column] == 'S' ||
    board[row][column] == 'D' || board[row][column] == 'P' || board[row][column] == 'F'){
        return false;
      }
    }
  }
  return true;
}
```

printBoard

```
void printBoard(char[][] board){
  console.println();
  console.println("The board looks like this: ");
  console.println();
  console.print (" ");
  for(int column = 0; column<10; column++){
    console.print(" " + column + " ");
  }
  console.println();
  for(int row = 0; row<10; row++){
    console.print(row + " ");
    for(int column = 0; column<10; column++){
      if(board[row][column] == '-'){
        console.print(" ");
      }
else if (board[row][column] == 'A' || board[row][column] == 'B' || board[row][column] =
board[row][column] == 'D' || board[row][column] == 'P' || board[row][column] == 'F'){
  console.print(" ");
}
else{
  console.print(board[row][column]);
}
if(column != 9){
   console.print(" | ");
}
}
console.println();
```

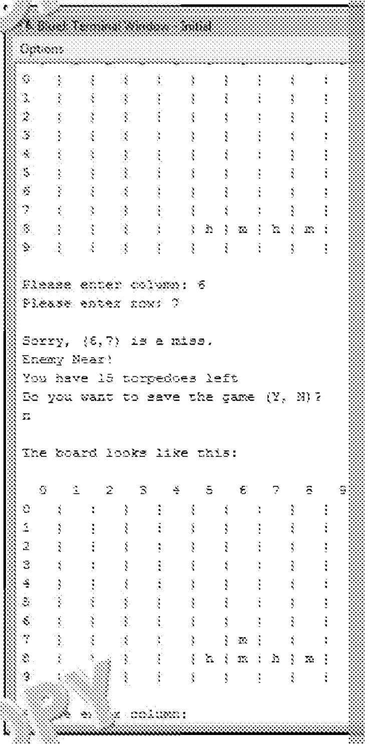


```
Computer placing the Battleship
Computer placing the Submarine
Computer placing the Destroyer
Computer placing the Patrol Boat
Computer placing the Frigate

The board looks like this:

  0   1   2   3   4   5   6   7   8   9
0 D |   |   |   |   | A | A | A | A | A
1 D |   |   |   |   |   |   |   |   |
2 D | B |   |   |   |   |   |   |   |
3   | B |   |   | P | P |   |   | S |
4   | B |   |   |   |   |   |   | S |
5   | B |   |   |   |   |   |   | S |
6   |   |   |   |   |   |   | F | F | F
7   |   |   |   |   |   |   |   |   |
8   |   |   |   |   |   |   |   |   |
9   |   |   |   |   |   |   |   |   |
MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
4. Board Test
9. Quit

Please enter your choice:
```

Question	Answer
10	

makePlayerMove

```
else if (board[row][column] == '-'){
    console.println("Sorry, (" + column + "," + row + ") is a miss.");
    board[row][column] = 'm';
    if (radarScan(board, row, column) == true)
    {
      console.println("Enemy Near!");
    }
    else

      console.println("All quiet");

  }
```



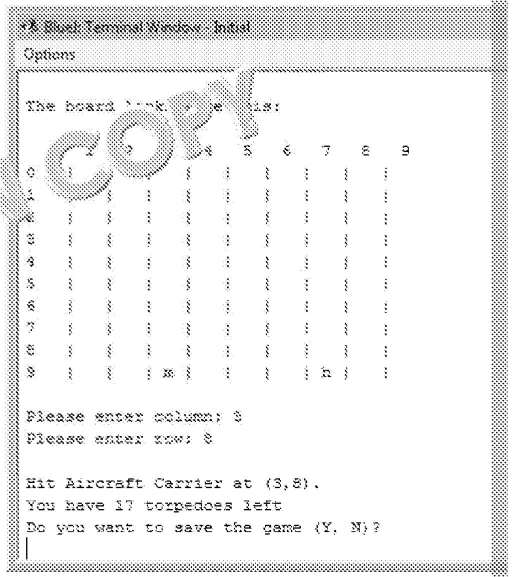
```
0    |   |   |   |   |   |   |   |   |
1    |   |   |   |   |   |   |   |   |
2    |   |   |   |   |   |   |   |   |
3    |   |   |   |   |   |   |   |   |
4    |   |   |   |   |   |   |   |   |
5    |   |   |   |   |   |   |   |   |
6    |   |   |   |   |   |   |   |   |
7    |   |   |   |   |   |   |   |   |
8    |   |   |   |   | h | m | h | m |
9    |   |   |   |   |   |   |   |   |

Please enter column: 6
Please enter row: 7

Sorry, (6,7) is a miss.
Enemy Near!
You have 15 torpedoes left
Do you want to save the game (Y, N)?
n

The board looks like this:

   0   1   2   3   4   5   6   7   8   9
0    |   |   |   |   |   |   |   |   |
1    |   |   |   |   |   |   |   |   |
2    |   |   |   |   |   |   |   |   |
3    |   |   |   |   |   |   |   |   |
4    |   |   |   |   |   |   |   |   |
5    |   |   |   |   |   |   |   |   |
6    |   |   |   |   |   |   |   |   |
7    |   |   |   |   |   | m |   |   |
8    |   |   |   |   | h | m | h | m |
9    |   |   |   |   |   |   |   |   |

Please enter column:
```

radarScan

```
boolean radarScan(char[][] board, int row, int column)
 {
    for (int x = column - 1; x < column + 2; x++)
    {
       for (int y = row - 1; y < row + 2; y++)
       {
          if ((x < 0) || (x > 9) || (y < 0) || (y > 9))
          {
             //off the board, so do nothing
          }
          else
          {
             if(board[y][x] != 'm' && board[y][x] != 'h' && board[y][x] != '-')
             {
                return true;
             }
          }
       }
    }
    return false;
 }
```

Question	Answer
11	*makePlayerMove*

```
...
else{
    String shipType = "";
    switch(board[row][column])
    {
      case 'A':
        shipType = "Aircraft Carrier";
        break;
      case 'B':
        shipType = "Battleship";
        break;
      case 'D':
        shipType = "Destroyer";
        break;
      case 'S':
        shipType = "Submarine";
        break;
      case 'P':
        shipType = "Patrol Boat";
        break;
    }
    console.println("Hit " + shipType + " at (" + column + "," + row + ").");
    board[row][column] = 'h';
  }
```



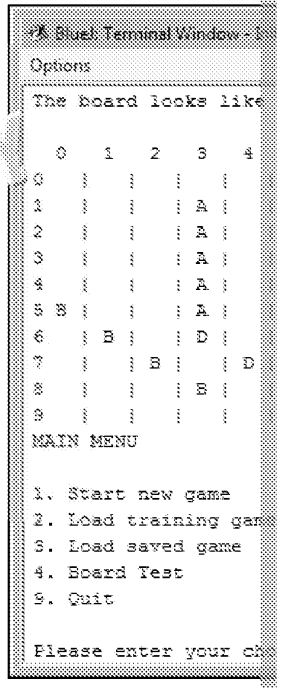
```
The board looks like this:

   0   1   2   3   4   5   6   7   8   9
0    |   |   |   |   |   |   |   |   |
1    |   |   |   |   |   |   |   |   |
2    |   |   |   |   |   |   |   |   |
3    |   |   |   |   |   |   |   |   |
4    |   |   |   |   |   |   |   |   |
5    |   |   |   |   |   |   |   |   |
6    |   |   |   |   |   |   |   |   |
7    |   |   |   |   |   |   |   |   |
8    |   |   |   |   |   |   |   |   |
9    |   |   | m |   |   |   | h |   |

Please enter column: 3
Please enter row: 8

Hit Aircraft Carrier at (3,8).
You have 17 torpedoes left
Do you want to save the game (Y, N)?
```

Question	Answer
12	*andomShips*

```
void placeRandomShips(char[][] board, Ship[] ships){
  char orientation = ' ';
  int row = 0;
  int column = 0;
  int HorV = 0;
  for(int i=0; i<ships.length; i++){
    boolean valid = false;
    while(!valid){
      row = random.nextInt(10);
      column = random.nextInt(10);
      HorV = random.nextInt(3);
      if(HorV == 0){
        orientation = 'v';
      }
      else if(HorV == 1){
        orientation = 'd';
      }
      else{
        orientation = 'h';
      }
      valid = validateBoatPosition(board, ships[i], row, column, orientation);
    }
    console.println("Computer placing the " + ships[i].name);
    placeShip(board, ships[i], row, column, orientation);
  }
}
```

validateBoatPosition

```
boolean validateBoatPosition(char[][] board, Ship ship, int row, int column, char orientatio
  if((orientation == 'v' || orientation == 'd')&& row + ship.size > 10){
    return false;
  }
  else if((orientation == 'h' || orientation == 'd')&& column + ship.size > 10){
    return false;
  }
  else{
    if(orientation == 'v'){
      for (int scan = 0; scan < ship.size; scan++){
        if (board[row + scan][column] != '-'){
          return false;
        }
      }
    }
    else if(orientation == 'h'){
      for (int scan = 0; scan < ship.size; scan++){
        if(board[row][column + scan] != '-'){
          return false;
        }
      }
    }
    else{
      for (int scan = 0; scan < ship.size; scan++){
        if(board[row + scan][column + scan] != '-'){
          return false;
        }
      }
    }
  }
  return true;
}
```



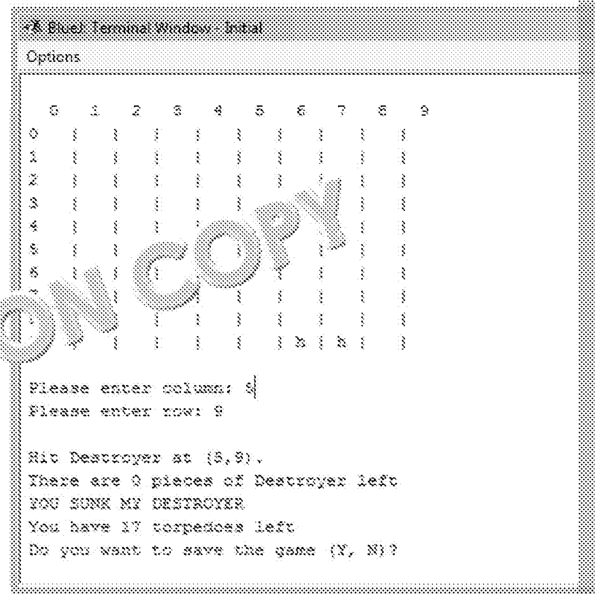
```
The board looks like

  0   1   2   3   4
0   |   |   |   |
1   |   |   | A |
2   |   |   | A |
3   |   |   | A |
4   |   |   | A |
5 B |   |   | A |
6   | B |   | D |
7   |   | B |   | D
8   |   |   | B |
9   |   |   |   |
MAIN MENU

1. Start new game
2. Load training gam
3. Load saved game
4. Board Test
9. Quit

Please enter your ch
```

placeShip

```
void placeShip(char[][] board, Ship ship, int row, int column, char orientation){
  if(orientation == 'v'){
    for (int scan = 0; scan < ship.size; scan++){
      board[row + scan][column] = ship.name.charAt(0);
    }
  }
  else if (orientation == 'h'){
    for (int scan = 0; scan < ship.size; scan++){
      board[row][column + scan] = ship.name.charAt(0);
    }
  }
  else{
    for (int scan = 0; scan < ship.size; scan++){
      board[row + scan][column + scan] = ship.name.charAt(0);
    }
  }
}
```

Question	Answer
13	*makePlayerMove*

```
else{
    int shipNum = -1;
    for (int x = 0; x < ships.length; x++)
    {
        if (board[row][column] == ships[x].name.charAt(0))
        {
            shipNum = x;
            ships[shipNum].size = ships[shipNum].size - 1;
        }
    }
    console.println("Hit " + ships[shipNum].name + " at (" + column + "," + row + ").");
    console.println("There are " + ships[shipNum].size + " pieces of " + ships[shipNum].nar
    if (ships[shipNum].size == 0)
    {
        console.println("YOU SUNK MY " + ships[shipNum].name.toUpperCase());
    }
    board[row][column] = 'h';
  }
```



BlueJ: Terminal Window - Initial

Options

```
   0   1   2   3   4   5   6   7   8   9
0    |   |   |   |   |   |   |   |   |
1    |   |   |   |   |   |   |   |   |
2    |   |   |   |   |   |   |   |   |
3    |   |   |   |   |   |   |   |   |
4    |   |   |   |   |   |   |   |   |
5    |   |   |   |   |   |   |   |   |
6    |   |   |   |   |   |   |   |   |
7    |   |   |   |   |   |   |   |   |
8    |   |   |   |   |   |   |   |   |
9    |   |   |   |   |   | h | h |   |

Please enter column: 5
Please enter row: 9

Hit Destroyer at (5,9).
There are 0 pieces of Destroyer left
YOU SUNK MY DESTROYER
You have 17 torpedoes left
Do you want to save the game (Y, N)?
```

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>

14

</td><td>

displayMenu

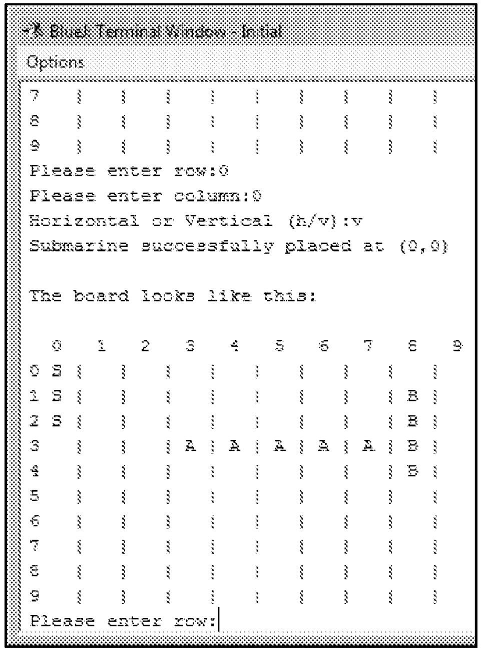
```
console.println("5. Manually place ships");
```

Main

```
if (menuOption == 5){
        placeManualShips(board, ships);
        playGame(board, ships);
    }
```

placeManualShips

```
void placeManualShips(char[][] board, Ship[] ships){
   char orientation = ' ';
   int row = 0;
   int column = 0;

   for(int i=0; i<ships.length; i++){
     boolean valid = false;
     while(!valid){
       row = console.readInteger("Please enter row:");
       column = console.readInteger("Please enter column:");
       orientation = console.readChar("Horizontal or Vertical (h/v):");
       valid = validateBoatPosition(board, ships[i], row, column, orientation);
       if (!valid)
       {
          console.println("Invalid location.  Please choose again.  ");
       }
     }
     console.println(ships[i].name +" successfully placed at ("+ row +","+column+")");
     placeShip(board, ships[i], row, column, orientation);
     realBoard(board);
   }
 }
```



```
7   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
8   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
9   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
Please enter row:0
Please enter column:0
Horizontal or Vertical (h/v):v
Submarine successfully placed at (0,0)

The board looks like this:

   0   1   2   3   4   5   6   7   8   9
0 S ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
1 S ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦ B ¦
2 S ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦ B ¦
3   ¦   ¦   ¦ A ¦ A ¦ A ¦ A ¦ A ¦ B ¦
4   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦ B ¦
5   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
6   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
7   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
8   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
9   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
Please enter row:
```

</td><td>

17 max

displayMenu

1 for added print statement in correct position

Main

1 for If statement, 1 for correct code inside; it is acceptable to not have created a separate method

placeManualShips

NOTE: this code can be largely copied and then amended from placeRandomShips.

1 for correct call statement

1 for setting up variables

1 for starting loop

1 for setting up flag

1 for starting While loop with correct condition

1 for prompting for row and column

1 for prompting for orientation

1 for executing validateBoatPosition

2 for correct invalid message

1 for correct print message

1 for placing the ship

1 for call to realBoard

1 for screen capture

</td></tr>
</table>

Question	Answer
15	

Score

```
class Score{
    String name;
    int score;
 }
```

setUpScores

```
void setUpScores(Score[] scores){
  scores[0] = new Score();
  scores[0].name = "George";
  scores[0].score = 17;
  scores[1] = new Score();
  scores[1].name = "Paul";
  scores[1].score = 19;
  scores[2] = new Score();
  scores[2].name = "John";
  scores[2].score = 23;
  scores[3] = new Score();
  scores[3].name = "Ringo";
  scores[3].score = 25;
  scores[4] = new Score();
  scores[4].name = "Bryan";
  scores[4].score = 35;
 }
```

displayMenu

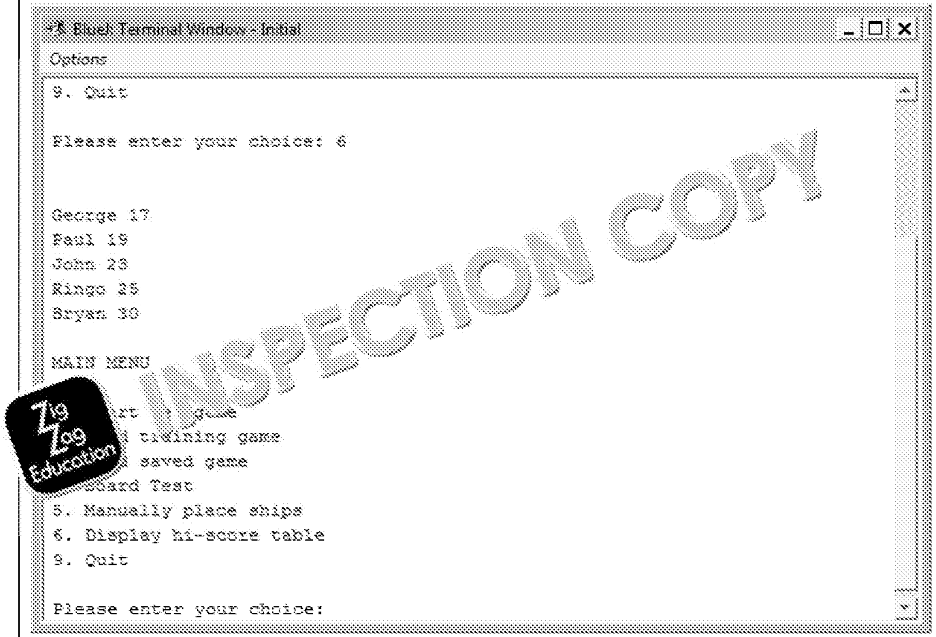
```
console.println("6. Display hi-score table");
```

displayHS

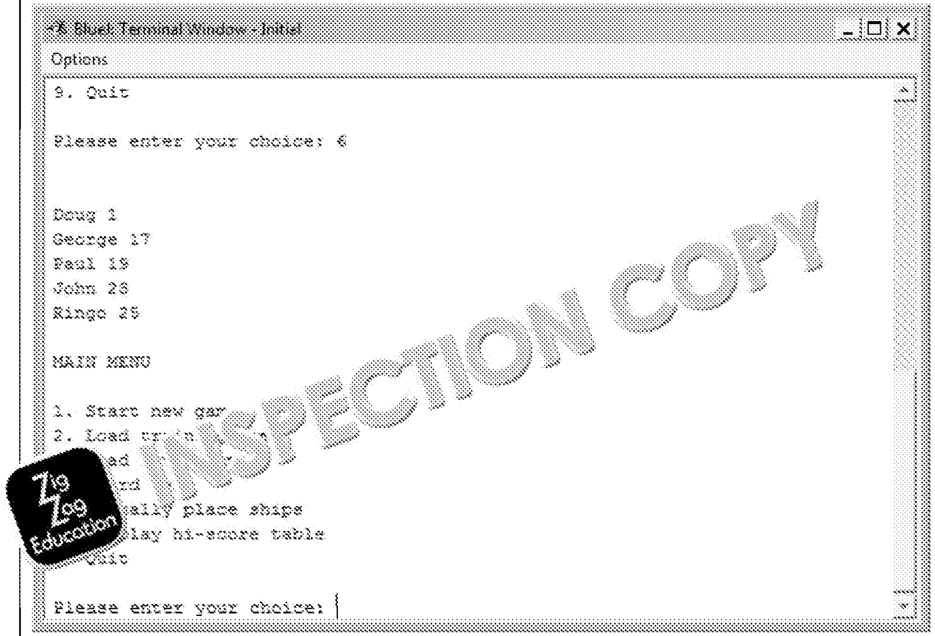
```
void displayHS(Score[] scores)
 {
   for (int index = 0; index < scores.length; index++)
   {
     console.println(scores[index].name + " " + scores[index].score);
   }
 }
```

	Main
	`if (menuOption == 6){`
	`        displayHS(scores);`
	`    }`
	
	`...`
	
	`public Main() {`
	`   final String TrainingGame = "Training.txt";`
	`   char board[][] = new char[10][10];`
	`   Ship ships[] = new Ship[5];`
	`   **Score scores[] = new Score[5];**`
	`   **setUpScores(scores);**`
	`   int menuOption = 0;`
	`   while(menuOption != 9){`
	`     setUpBoard(board);`
	`     setUpShips(ships);`
	`     displayMenu();`
	`     menuOption = GetMainMenuChoice();`
	`     if (menuOption == 1){`
	`       placeRandomShips(board, ships);`
	`       playGame(board, ships, **scores**);`
	`     }`
	`     if (menuOption == 2){`
	`       loadGame(TrainingGame, board);`
	`       playGame(board, ships, **scores**);`
	`     }`
	
	playGame
	`void playGame(char[][] board, Ship[] ships, **Score[] scores**){`
	
	`...`
	
	`**int score = 0;**`
	`   while(!GameWon && torpedoes > 0){`
	`     printBoard(board);`
	`     makePlayerMove(board, ships);`
	`     **score = score + 1;**`
	
	`...`

```
if(GameWon){
        console.println("All ships sunk!");
        console.println();
        if (score < scores[4].score)
        {
          scores[4].score = score;
          console.println("Well done, you got a high score");
          scores[4].name = console.readLine("Enter your name: ");
          bubSortScores(scores);
        }
    }

  Sco[illegible]
  [illegible] bubSortScores(Score[] scores)
  {
    boolean flag = true;
    Score temp;
    while(flag)
    {
      flag = false;
      for(int j=0;  j < scores.length - 1;  j++ )
      {
        if (scores[j].score > scores[j+1].score )
        {
          temp = scores[j];
          scores[j] = scores[j+1];
          scores[j+1] = temp;
          flag = true;
        }
      }
    }
  }
```



```
9. Quit

Please enter your choice: 6


George 17
Paul 19
John 23
Ringo 25
Bryan 30

MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
4. Board Test
5. Manually place ships
6. Display hi-score table
9. Quit

Please enter your choice:
```



```
9. Quit

Please enter your choice: 6


Doug 1
George 17
Paul 19
John 23
Ringo 25

MAIN MENU

1. Start new game
2. Load training game
3. Load saved game
4. Board Test
5. Manually place ships
6. Display hi-score table
9. Quit

Please enter your choice:
```

Further Modifications

Ideas for modifications	How to in

Name	

ZigZag Education supporting

AS AQA Computer Science Paper 1

Summer 2016: AQA WARSHIPS

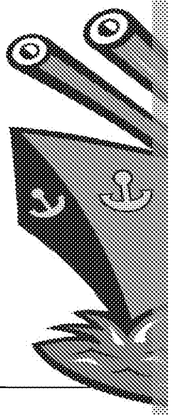
Electronic Answer Document (EAD)

Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and any additional pages

- Answer **all** questions
- The marks available for each question are shown in brackets

- You will need:
  - □ access to a computer
  - □ access to a printer
  - □ access to appropriate software
  - □ electronic copies of the required skeleton code
  - □ EAD (Electronic Answer Document)



Total marks:

Programming Theory Questio

Answer all questions.
Remember to save this document regularly.

Q		Answer
1	(a)	
	(b)	
	(c)	
	(d)	
	(e)	
2		
3		
4		
5		
6		
7	(a)	
	(b)	
8	(a)	
	(b)	
	(c)	
	(d)	
	(e)	
9	(a)	
	(b)	
10		
11		
12		
13		
14		
15	(a)	
	(b)	

Programming Exercises

Answer all questions.
Remember to save this document regularly.

Q	Answer
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	