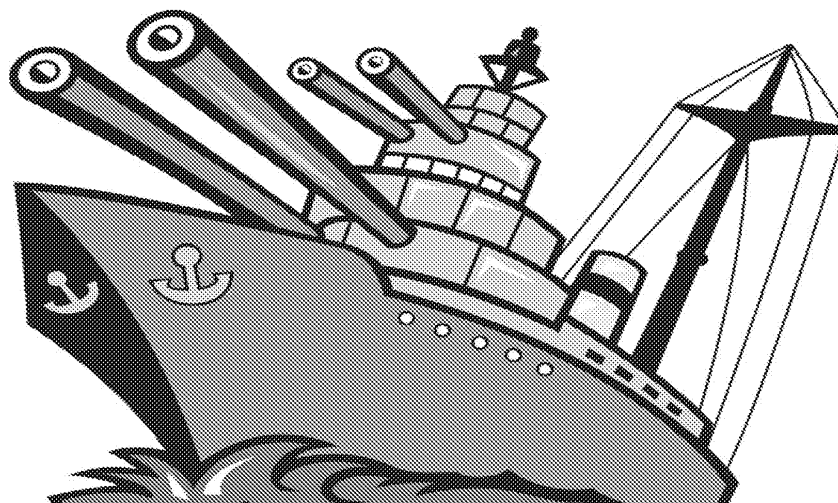


2015 specification
for the 2016 AS exam

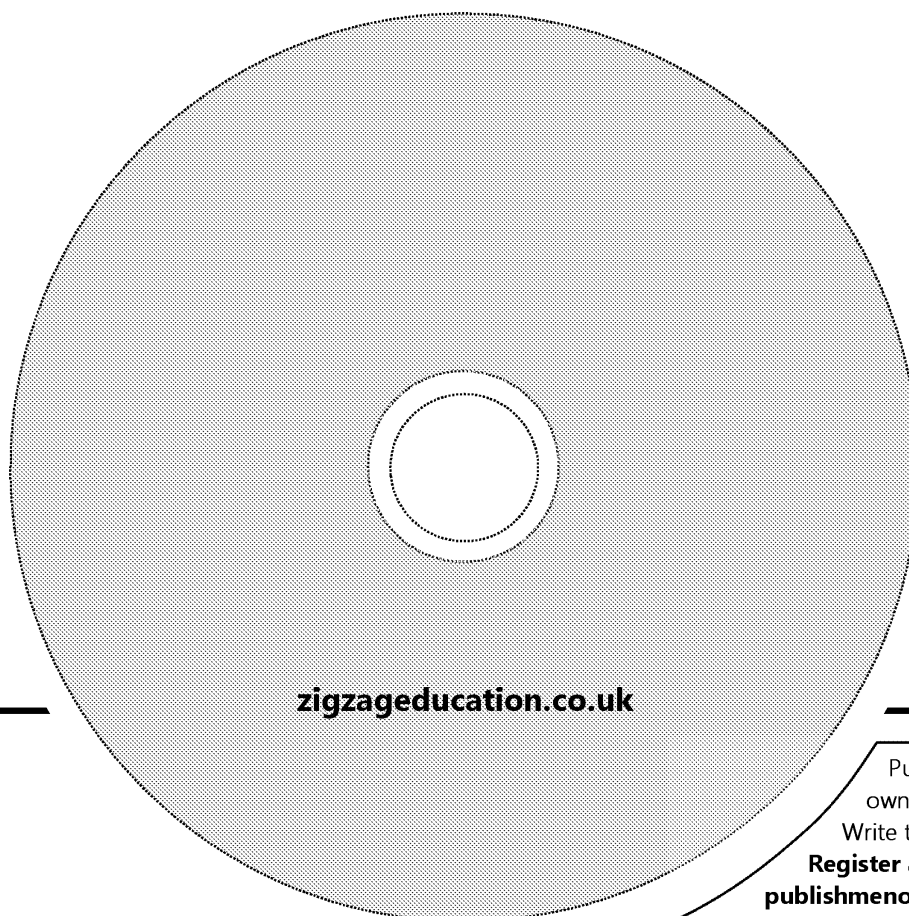


PAPER 1 EXAM RESOURCE PACK 2016

AQA WARSHIPS

for AS AQA Computer Science

PASCAL



**AS2/
6500**

**POD
6501**

Publish your
own work...
Write to a brief...

Register at
publishmenow.co.uk

Contents

Thank You for Choosing ZigZag Education	ii
Teacher Feedback Opportunity	iii
Terms and Conditions of Use	iv
Teacher's Introduction	1
Programming Exercises: Teacher Notes	2
Suggested Question Combinations	2
Possible Additional Questions	2
Pre-Release Commentary	3
Description of the Program	3
Description of Program Elements	4
Description of Program Routines	5
Structure Chart Activity	9
Programming Theory Questions	10
Write-on format	10
Non- write-on format	14
Programming Exercises	16
Answers and Solutions	21
Structure Chart (Solution)	21
Programming Theory Questions (Answers)	22
Programming Exercises (Solutions)	23
Appendices	40
Further Modifications	40
Electronic Answer Document (EAD) Printout	41

Teacher's Introduction

This pack is designed to help you support your students taking the AQA Computer Science AS exam. It is based on the AQA Paper 1 'AQA Warships' preliminary material (PASCAL) – for

① Pre-release Commentary (for teachers)

A detailed overview of the skeleton program, describing all PASCAL code.

This section is designed to help you get to grips with the program, so that you can support your students. This commentary is not designed to be given to students before they have worked on 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 program. Any missing routines and variables must be added to the diagram. A complete solutions section is provided at the end of the resource.

③ Programming Theory Questions

Theory questions test students' understanding of the 'AQA Warships' code. These are provided in both write-on and non-write-on format.

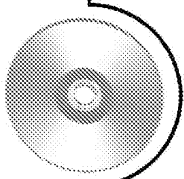
④ Programming Exercises

Modification exercises put students' programming skills to the test, like Section 4. An Electronic Answer Document (EAD) and the modified PASCAL code are provided.

Answers and solutions for the structure chart activity, theory questions and programming exercises from page 21 onwards. Note that for the programming exercises in particular, the marks are subjective, so 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 program. This is suggested as a group activity, so that students (and the teacher) can brainstorm together, 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 provided on the CD.



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

- **MODIFIED_PASCAL_CODE.txt** – text file containing the additional PASCAL code as shown in the mark scheme for section ④ (from page 21 onwards).
- **PAPER1_EAD.docx** – Electronic Answer Document for completion of the programming exercises.

This resource is intended to support your teaching only. It is the teacher's responsibility to ensure that this resource is used appropriately. You must ensure that you use this resource to support your teaching and to help you prepare your lessons. You must also consider whether it is appropriate to hand out some of the activities for classwork or homework. You may also consider hand out the booklet to be worked through by your students more independently. If you use this material, it is the teacher's responsibility to decide in what way to assist the students. 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. We do not have any special knowledge of what to expect on any particular exam.

INSPECTION COPY

COPYRIGHT
PROTECTED



Suggested Question Combinations

It is not envisaged that a student would complete all questions in a 1-hour period. One approach is to get students to work through all the questions under 'open-book' conditions, and then the questions can be followed up by setting combinations of the questions under test conditions similar to the following:

- 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 & Q3	25 minutes	Q8 & Q12	30 minutes
Q3, Q5, Q6 & Q7	30 minutes	Q13 & Q15	60 minutes
Q8 & Q9	20 minutes	Q8 & Q14	35 minutes
Q10 & Q11	25 minutes		

It is also useful (and fun) to get students to come out and solve a question 'live' at the front of the class with their classmates.

Possible Additional Questions

1. When the game has finished, tell the user how accurate they were as a percentage of hits by the total number of shots. E.g. 10 hits, 30 shots = 33% hit rate. Only display the percentage if it is not 0%.
2. One shot sinks a ship.
3. Sea mine is placed on the board. If the player hits it, they lose and the game ends.
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 temporarily reveals the positions of all ships.
9. Add an ammo store to the board. If the player hits it they get 10 more torpedoes.
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 the message "[name] won".
13. Allow the 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 the torpedoes.
16. Adapt the missile task (above) so that the user can choose whether to use a missile or a torpedo. The user can fire a maximum of 2 missiles.
17. Add a main menu option which will allow you to select which ships are to be placed on the board.
18. Enhance the computer player in task 15 further so that if it hits a square it will continue to fire until a ship is sunk.

INSPECTION COPY

COPYRIGHT
PROTECTED



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 grid. The player takes shots at different column (0—9) and a row (0—9).

The ships 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 of the board. This is then populated into Board (a two-dimensional array of Chars). The 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 hit).

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

The game proceeds by asking the player for a column and a row. The program checks if this index in the Board array. If it is a — this is replaced by an m. If it is a letter, this is replaced by an h. If this position already contains an m or an h, a message is displayed.

If a position on 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 on the number of shots a player may take. The player can keep firing until they have hit every square.

INSPECTION COPY

COPYRIGHT
PROTECTED



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
TShip	User-defined type from which the ship variable is defined.	Declares a type of Record to contain data for a ship.
Name	A string type within a record structure.	String type to store the ship's name as a constant.
Size	An integer type within a record structure.	Integer type to store the ship's length as a constant.
TBoard	User-defined type from which the Board variable is defined.	Declares a type of 2 dimensional array of characters.
Ship	User-defined variable 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 ships
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 has chosen
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 char variable	Stores direction of a ship: V for vertical, H for horizontal
HorV	An integer variable	Used to randomly generate the orientation

INSPECTION COPY

**COPYRIGHT
PROTECTED**



Description of Program Routines

The program functions (F) and procedures (P) are described below.

Routine	Description
CheckWin (F)	<p>Receives: Board Returns: Boolean Called from: PlayGame</p> <p>Checks every position in board. Returns False if it finds a player win. Returns True if it checks every position and no player has won.</p>
DisplayMenu (P)	<p>Receives: nothing Returns: nothing Called from: main program</p> <p>A simple procedure that prints the menu.</p>
GetMainMenuChoice (F)	<p>Receives: nothing Returns: integer Called from: main program</p> <p>Handles the user's menu choice.</p> <ol style="list-style-type: none"> 1. Prompts the user to enter a choice. 2. Returns that number.
GetRowColumn (P)	<p>Receives: Row (by reference), Column (by reference) Returns: nothing Called from: MakePlayerMove</p> <ol style="list-style-type: none"> 1. Prompts the user for row. 2. Prompts the user for column. 3. Returns Row and Column.
LoadGame (P)	<p>Receives: FileName, Board (by reference) Returns: nothing Called from: main program</p> <ol style="list-style-type: none"> 1. Links the data context to the file. 2. Reads in a row of data. 3. Then chops that line into pieces. 4. Adds the pieces to the board. Board updated. 5. Repeats for all 10 rows. 6. Closes the file.
MakePlayerMove (P)	<p>Receives: Board (by reference), Ship (by reference) Returns: nothing Called from: PlayGame</p> <ol style="list-style-type: none"> 1. Receives the row and column. 2. Splits them into separate variables. 3. Checks whether the ship is in the row. 4. Checks whether the ship is in the column. 5. If neither 3 nor 4 are true, then the ship is not in the row or column.

INSPECTION COPY

COPYRIGHT
PROTECTED



Description

Receives: Board (by reference), 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. Board update by reference.

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.Size). 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.

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 While 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, the value of GameWon will now be True and the loop will exit after this iteration)
 - 2.4. Prints a success message if the game has been won

**COPYRIGHT
PROTECTED**



Routine	Description
PrintBoard (P)	<p>Receives: Board Returns: nothing Called from: PlayGame</p> <p>Displays the board:</p> <ol style="list-style-type: none"> 1. Starts off by displaying a message. 2. A For loop is used to print the board. 3. Nested For loops now display the board. <ol style="list-style-type: none"> 3.1. Prints the row number 3.2. Second For loop works <ol style="list-style-type: none"> 3.2.1. An empty square is displayed 3.2.2. A square with ship is displayed 3.2.3. Anything else (a hit) 3.2.4. A separator is displayed
SetUpBoard	<p>Receives: Board (by reference) Returns: nothing Called from: main program</p> <ol style="list-style-type: none"> 1. Cycles through all positions on the board. <ol style="list-style-type: none"> 1.1. Assigns all positions on the board to be empty. <p>Some of these dashes will be replaced by ships. Board updated by reference.</p>
SetUpShips (P)	<p>Receives: Ships (by reference) Returns: nothing Called from: main program</p> <p>Initialises the ships in the array (by reference). Sets the name of each ship. Sets the orientation. Ships updated by reference.</p>
ValidateBoatPosition (F)	<p>Receives: Board, Ship, Row, Column, Orientation Returns: Boolean Called from: PlaceRandomShips</p> <p>Checks to see whether it is possible to place the boat on the board. Does the boat run off the edge of the board?</p> <ol style="list-style-type: none"> 1. If the row number plus the ship's length is greater than the number of rows on the board, then it will go off the edge of the board. 2. If the column number plus the ship's length is greater than the number of columns on the board, then it will go off the edge of the board. 3. If the ship is vertical: <ol style="list-style-type: none"> 3.1. A For loop scans along the row. <ol style="list-style-type: none"> 3.1.1. If a position isn't empty, then it's not possible to place the ship. 4. If the ship is horizontal: <ol style="list-style-type: none"> 4.1. A For loop scans along the column. <ol style="list-style-type: none"> 4.1.1. If a position isn't empty, then it's not possible to place the ship. 5. If this part of the function is successful, then True is returned.

INSPECTION COPY

COPYRIGHT
PROTECTED



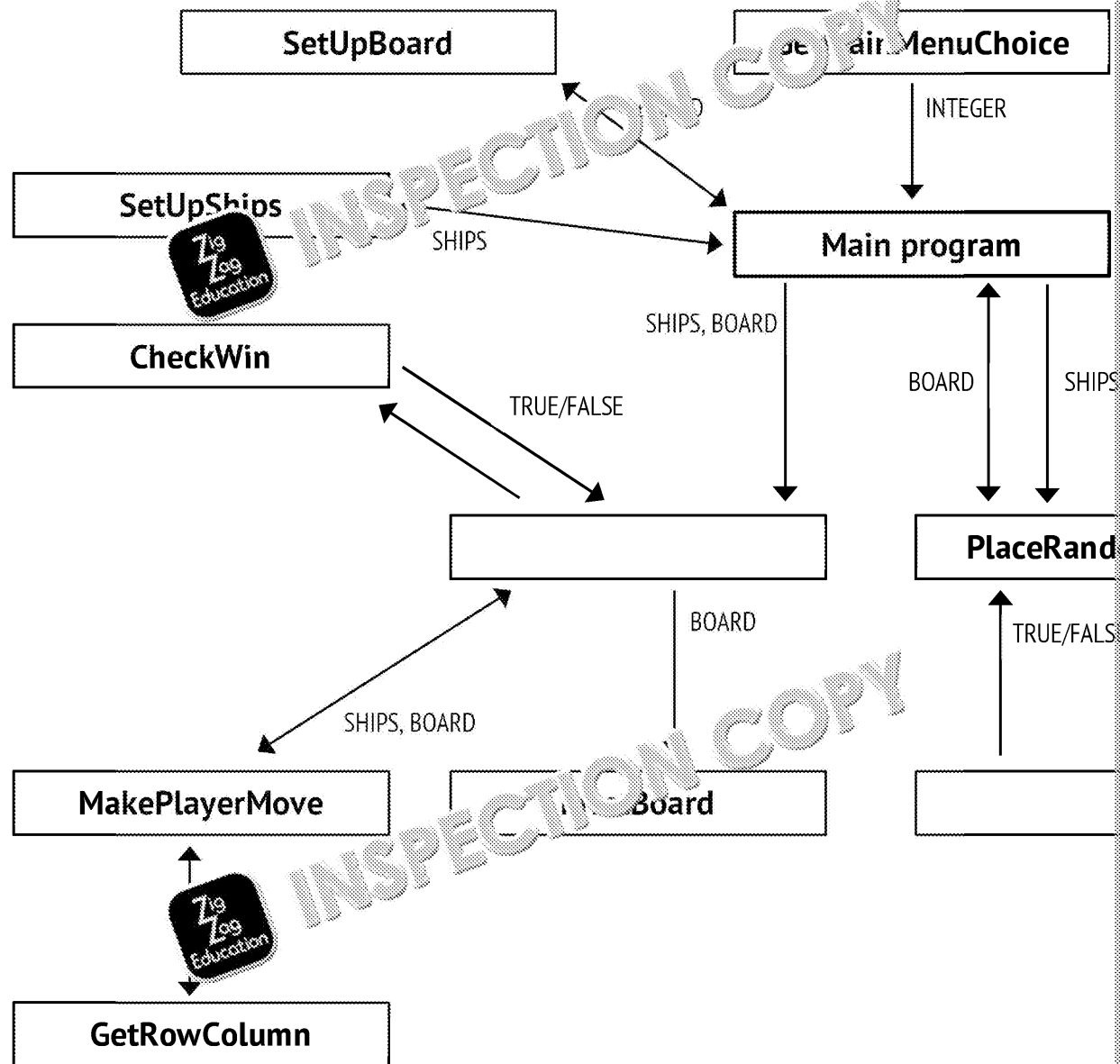
Routine	Description
Main program	<ol style="list-style-type: none"> 1. The variable MenuOption stores the menu option has been selected (wasn't 9) 2. Starts a While loop that continues until the user selects option 9 (to exit) <ol style="list-style-type: none"> 2.1. Populates Board with data by calling SetUpBoard (this would read from a text file) 2.2. Populates Ships with data by calling SetUpShips 2.3. Displays the board by calling DisplayMenu 2.4. Calls the menu choice to get the user's choice and stores it in the variable MenuOption 2.5. If the user picks option 1: <ol style="list-style-type: none"> 2.5.1. The board is populated by the ships in random locations 2.5.2. The game is started 2.6. If the user picks option 2: <ol style="list-style-type: none"> 2.6.1. The board is populated from the training text file 3. The game is started
Var Const Type definitions	<ol style="list-style-type: none"> 1. Sets the constant TrainingGame to the correct filename; the default is 'training.txt' 2. Declares (creates) a two-dimensional array of String type TBoard, to store the board 3. Declares a record structure for TShip to be used to create variables to store ship data 4. Declares (creates) an array of ships type to be used to create variables to store ship data

INSPECTION COPY

COPYRIGHT
PROTECTED



AQA WARSHIPS



INSPECTION COPY

COPYRIGHT
PROTECTED



Programming Theory Questions

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 function `ValidateBoatPosition`.

What is the purpose of the variable `orientation`?

.....

.....

.....

3. What data is stored for each ship?

.....

.....

.....

4. Look at the procedure `isInLine`.

What is the purpose of the `While` loop?

.....

.....

.....

.....

INSPECTION COPY

COPYRIGHT
PROTECTED



5. The subroutine CheckWin uses the code `Board[Row][Column]`, with two sets of square brackets, whereas the subroutine SetUpShips uses the code `Ships[1]` with just one set of square brackets. Explain the difference between using one set of square brackets compared with two.

6. Explain the operation of the procedure 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 value

**COPYRIGHT
PROTECTED**



9. Look at the procedure 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 LoadGame. These routines both use a local variable called Row. What are local variables? To these routines what is an advantage of utilising local variables?



INSPECTION COPY

.....

.....

.....

.....

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?

.....

.....

.....

.....

.....

12. PrintBoard is a procedure, whereas MainMenuChoice is a function. Describe the difference between a procedure and a function.



INSPECTION COPY

.....

.....

.....

**COPYRIGHT
PROTECTED**



13. What is the purpose of the following line?

```
CloseFile(CurrentFile);
```

INSPECTION COPY

14. What is the purpose of these lines?

```
ReadIn(CurrentFile, Line);  
For Column := 0 To 9 Do  
  Board[Row][Column] := Line[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.



**COPYRIGHT
PROTECTED**



Programming Theory Questions

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 code in the procedure `validateBoatPosition`.
What is the purpose of the variable `orientation`?
3. What data is stored for each ship?
4. Look at the procedure `PlayGame`.
What is the purpose of the `While` loop?
5. The subroutine `CheckWin` uses the code `Board[Row][Column]`, with two sets of square brackets, whereas the subroutine `SetUpShips` uses the code `Ships[1]` with just one set of square brackets.
Explain the difference between using one set of square brackets compared to two.
6. Explain the operation of the procedure `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 `repeat` 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 value
9. Look at the procedure `PrintBoard`.
 - (a) What lines of code print the column headings?
 - (b) What is the advantage of this method over 'hard-coding'?

INSPECTION COPY

COPYRIGHT
PROTECTED



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? To these routines what is an advantage of utilising local variables?

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?

12. PrintBoard 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?

```
CloseFile(CurrentFile);
```

14. What is the purpose of these lines?

```
ReadLn(CurrentFile, Line);
For Column := 0 To 9 Do
  Board[Row][Column] := Line[Column + 1];
```

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

- What would happen to the program if Training.txt did not exist?
- Describe how we would change the program to solve this.

**COPYRIGHT
PROTECTED**



Programming Exercises

The following require you to open the skeleton program and make modifications. They also require you to test your code 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. You need to modify the program so 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. Although the game is designed to last 20 turns, only has 20 torpedoes. The number of torpedoes should decrease by 1 after every shot. When the number of torpedoes reaches 0, the message 'GAME OVER! You have lost the game.' 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 to 0 and the 'GAME OVER! You have lost the game.' message

Question 3

This question refers to DisplayMenu and the main program.

Alter the menu so that 'OPTION 3' is also displayed between options 2 and 9.

The menu should now 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 the main program
- SCREEN CAPTURE(S) of testing

INSPECTION COPY

COPYRIGHT
PROTECTED



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 ready?' message. 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

Question 5

This question refers to the main program.

Option 3 currently just displays a message. Amend it so that it prompts the user to enter a filename, opens 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'

Question 6

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

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

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 the game? (Y/N)'. If the player enters Y, they should then be prompted for a filename and the game state should be saved to a file created in the current directory.

Evidence you need to provide

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

INSPECTION COPY

COPYRIGHT
PROTECTED



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 displayed, return to the main menu. A procedure called RealBoard (similar to PrintBoard) sets up the board.

Evidence you need to provide

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

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 code to place it in addition to the original ships when option 1 or 4 is selected. 'F' will represent the 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

Question 10

This question refers to MakePlayerMove.

When a player misses, a radar scan of the adjacent cells should be performed. If a section of ship is detected, the message 'Enemy Near!' should be displayed. If not, the message 'No Enemy Near!' should be displayed. You should create a function called RadarScan that returns a Boolean (true if enemy near, false if not).

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

INSPECTION COPY

COPYRIGHT
PROTECTED



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 SOURCE CODE PROGRAM for PlayGame
- 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 left on 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 ship placed diagonally down and to the left

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 appropriate message should say that the ship has been sunk.

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

Evidence you need to provide

- Your amended SOURCE CODE PROGRAM for MakePlayerMove
- SCREEN CAPTURE(S) of a ship being sunk

INSPECTION COPY

COPYRIGHT
PROTECTED



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. The program will then check whether this location is valid using `ValidateBoatPosition`. If valid, 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 `RealTime` procedure should display the position of the 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.

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 10 points for every ship sunk. A higher 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` and `SetUpShips`) called `SetUpScores`. It should take 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 to display the high score table.

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

If a player's score is less than somebody on the table (remember that a lower score is better), their score should be replaced with their name (you will need to prompt for this) and the table should be re-sorted using `BubSortScores`.

Evidence you need to provide

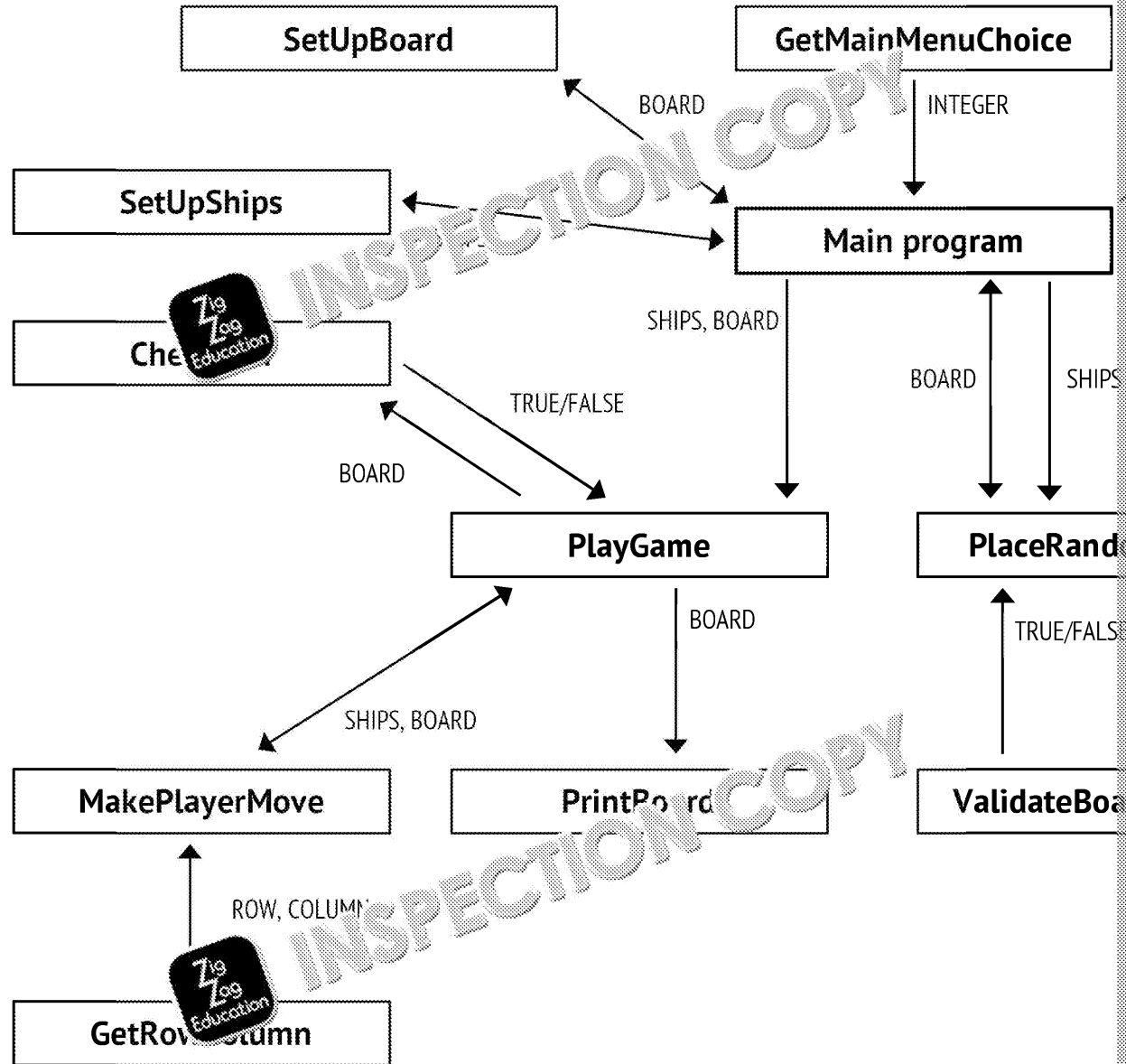
- Your amended sections of the SOURCE CODE PROGRAM highlighting your changes.
- SCREEN CAPTURE(S) showing the table being displayed before and after a player's score is added.

INSPECTION COPY

COPYRIGHT
PROTECTED



Structure Chart (Solution)



INSPECTION COPY

COPYRIGHT
PROTECTED



Programming Theory Questions (Answers)

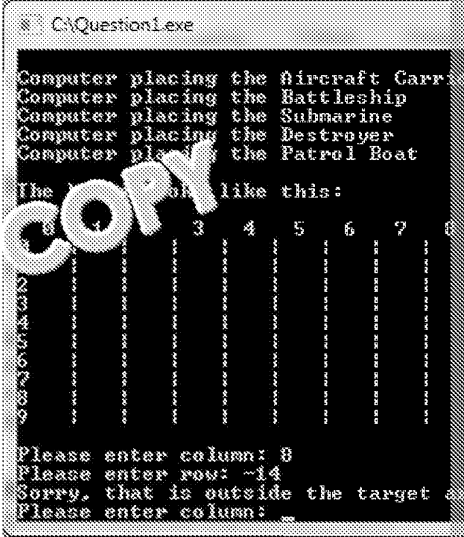
Q	Marking Guidance
1a	Ships / Board
1b	ValidateBoatPosition
1c	Row / Column / HorV / MenuOption
1d	GetRowColumn / ValidateBoatPosition / CheckWin / GetMainMenuChoice
1e	Valid / GameWon
2	To store whether the boat should be vertically or horizontally positioned (1 mark) board (1 mark)
3	Name (1 mark), Size (1 mark)
4	To ensure that the board is reprinted (1 mark) and the user input requested again (1 mark) while the game is not over (1 mark).
5	One set of brackets is used with a one-dimensional array (1 mark), two sets of two-dimensional array (1 mark).
6	To check whether the ship can be placed on the board (1 mark) by ensuring the 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
7a	Character array / char array / 2D array of characters
7b	Two-dimensional array (1 mark) / 10-by-10 array (1 mark) / One dimension for the row (1 mark) / A row,column / x,y value is used to refer
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 Column := 0 To 9 Do Write(' ', Column, ' ');
9b	It is easier to modify the game (1 mark), it allows many lines of code to be combined (1 mark).
10	Local variable: stores a value for only that particular routine. The value is lost when the routine exits (1 mark). Both routines can use the <u>same variable names</u> to traverse the array <u>without a loop</u> (2 marks for showing understanding of underline words, 1 mark for partial understanding)
11	A For loop repeats a set number of times (1 mark) and the number of times is known before the loop starts (1 mark). A While loop repeats an unknown number of times (1 mark) while a certain condition is met
12	A procedure is a routine called by the program which performs a set of actions (1 mark). A function is a routine called within an expression which returns a result (1 mark).
13	The file must be closed after it has been used or it cannot be accessed by other programs
14	Reads a line of the training game file (1 mark), then for each column (1 mark) of the line, it reads individual characters (1 mark) and assigns them to the correct position on the board
15a	It would crash
15b	A try catch should be used to catch the error or detect the presence of the file (1 mark), then call the file function (2 marks) and then display a suitable error message (1 mark)

INSPECTION COPY

**COPYRIGHT
PROTECTED**



Programming Exercises (Solutions)

Question	Answer
1	<p><u>GetRowColumn:</u></p> <pre>Procedure GetRowColumn(Var Row : Integer; Var Column : Integer) Begin Writeln; Write('Please enter column: '); Readln(Column); Write('Please enter row: '); Readln(Row); If (row > 9) Or (row < 0) Or (column > 9) Or (column < 0) Do Begin Writeln('Sorry, that is outside the target area. Please select again'); Write('Please enter column: '); Readln(Column); Write('Please enter row: '); Readln(Row); End; Writeln; End;</pre> <div></div>

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
2	<p>PlayGame</p> <pre> Procedure PlayGame(Board : TBoard; Ships : TShips); Var GameWon : Boolean; Torpedoes : Integer; Begin Torpedoes := 20; GameWon := False; While (True) And (Torpedoes > 0) Do Begin PrintBoard(Board); MakePlayerMove(Board, Ships); Torpedoes := Torpedoes - 1; Writeln('You have ', Torpedoes, ' torpedoes left'); GameWon := CheckWin(Board); If GameWon = True Then Begin Writeln('All ships sunk!'); Writeln; End; If Torpedoes = 0 Then Begin Writeln('GAME OVER! You ran out of a torpedoes'); Writeln; End; End; End; </pre>

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
4	<p><u>Main program</u></p> <pre> Var Board : TBoard; MenuOption : Integer; Ships : Tships; Sure : Char; End // ; removed from the program Else If MenuOption = 3 Then Begin WriteLn('Are you sure (Y,N) ?'); ReadLn(Sure); If Sure <> 'Y' Then MenuOption := 0; End; End; End.</pre> <p><u>Declarations</u></p> <pre> Var Board : TBoard; MenuOption : Integer; Ships : Tships; Sure : Char; FileName : String;</pre> <p><u>Main program</u></p> <pre> Else If MenuOption = 3 Then Begin WriteLn('Enter filename:'); ReadLn(FileName); LoadGame(FileName, Board); PlayGame(Board, Ships); End;</pre>
5	<p><u>Declarations</u></p> <pre> Var Board : TBoard; MenuOption : Integer; Ships : Tships; Sure : Char; FileName : String;</pre> <p><u>Main program</u></p> <pre> Else If MenuOption = 3 Then Begin WriteLn('Enter filename:'); ReadLn(FileName); LoadGame(FileName, Board); PlayGame(Board, Ships); End;</pre>

```

C:\Question5.exe
MAIN MENU
1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice:
Are you sure (Y,N) ?
MAIN MENU
1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice:
Y
```

```

C:\Question5.exe
1. Start new game
2. Load training game
3. Load saved game
9. Quit

Please enter your choice:
Enter filename:
training.txt

The board looks like:

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

Please enter color:

```

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
6	<p><u>SaveGame</u> Procedure SaveGame(Filename:String; Board:TBoard); Var CurrentFile : Text; Column : Integer; Row : Integer; Line: String; Begin AssignFile(CurrentFile, FileName); Rewrite(CurrentFile); For Row := 0 To 9 Do Begin Line := ''; For Column := 0 To 9 Do Line := Line + Board[Row][Column]; Writeln(CurrentFile, Line); End; CloseFile(CurrentFile); End;</p>
7	<p><u>PlayGame</u> Procedure PlayGame(Board : TBoard; Ships : TShips); Var GameWon : Boolean; Torpedoes : Integer; Yn : Char; Filename : String; Writeln('All ships sunk!'); Writeln; End // ; removed here Else Begin Writeln('Do you want to save? (Y, N)?'); Readln(Yn); If Yn='Y' Then Begin Writeln('Enter filename:'); Readln(Filename); SaveGame(Filename, Board); End; End; If Torpedoes = 0 Then ...</p>

```

C:\Question7.exe
1. Start new game
2. Load training
3. Load saved game
9. Quit

Please enter your choice:
Enter filename:
test.txt

The board looks like:
0 1 2 3
0 0 0 0
1 0 0 0
2 0 0 0
3 0 0 0
4 0 0 0
5 0 0 0
6 0 0 0
7 0 0 0
8 0 0 0
9 0 0 0

Please enter column:

```

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
8	<p>DisplayMenu</p> <pre>Writeln('4. Board Test');</pre> <p>Main program</p> <pre>Else If MenuOption = 4 Then Begin PlaceRandomShips(Board, Ships); RealBoard(Board); End;</pre> <p>RealBoard</p> <pre>Procedure RealBoard(Board : TBoard); Row, Column : Integer; begin Writeln; Writeln('The board looks like this: '); Writeln; For Column := 0 To 9 Do Begin Write(' ', Column, ' '); End; Writeln; For Row := 0 To 9 Do Begin Write(Row, ' '); For Column := 0 To 9 Do Begin If Board[Row][Column] = '-' Then Write(' '); Else If (Board[Row][Column] = 'A') Or (Board[Row][Column] = 'B') Or (Board[Row][Column] = 'L') Or (Board[Row][Column] = 'P') Then Write(' '); Else Write(Board[Row][Column]); End; If (Column <> 9) Then Write(' '); End; Writeln; End; End;</pre>

```

C:\Question8.exe
Computer placing
Computer placing
The board looks like this:
 0  1  2  3
0 D  D  D
1
2
3
4
5
6  S  S  S
7
8
9
MAIN MENU
1. Start new game
2. Load training
3. Load saved game
4. Board Test
7. Quit
Please enter your
  
```

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
9	<p><u>SetUpShips</u></p> <p>Ships[5].Name := 'Frigate'; Ships[5].Size := 3;</p> <p><u>Declarations</u></p> <p>TShips = Array[0..5] Of Tship</p> <p><u>CheckWin</u></p> <p>GameWon := True; For Row := 0 To 9 Do For Column := 0 To 9 Do If (Board[Row][Column] = 'A') Or (Board[Row][Column] = 'B') Or (Board[Row][Column] = 'S') Or (Board[Row][Column] = 'D') Or (Board[Row][Column] = 'P') Or (Board[Row][Column] = 'F') Then GameWon := False; CheckWin := GameWon; End;</p> <p><u>PrintBoard</u></p> <p>... For Column := 0 To 9 Do Begin If Board[Row][Column] = '-' Then Write(' ') Else If (Board[Row][Column] = 'A') Or (Board[Row][Column] = 'B') Or (Board[Row][Column] = 'S') Or (Board[Row][Column] = 'D') Or (Board[Row][Column] = 'P') Or (Board[Row][Column] = 'F') Then Write(' ') Else Write(Board[Row][Column]); End; ... WriteLn;</p> <p><u>PlaceRandomShips</u></p> <p>For Count := 0 To 5 Do</p>

```

C:\Quest
Computer
Computer
The board
  0  1
0  :  :
1  D  :
2  D  :
3  D  :
4  :  :
5  :  :
6  B  B
7  :  :
8  :  :
9  :  :
MAIN MENU
1. Start
2. Load
3. Load
4. Board
9. Quit
Please en

```

INSPECTION COPY

COPYRIGHT
PROTECTED



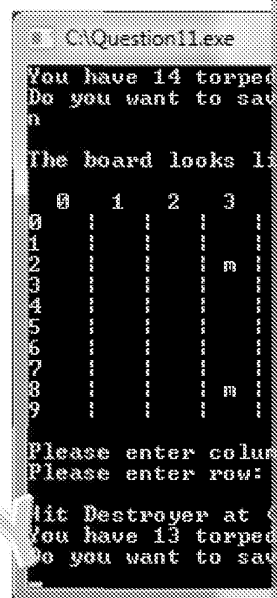
Question	Answer
10	<p><u>MakePlayerMove</u></p> <pre> Else If Board[Row][Column] = '-' Then Begin Writeln('Sorry, (', Column, ', ', Row, ') is a miss.');</pre> <p>Board[Row][Column] := 'm';</p> <pre> If RadarScan(Board, Row, Column) = True Then Writeln('Enemy Near!') Else Writeln('All good.')</pre> <p>End;</p> <p>Function RadarScan(Board : TBoard; Row : Integer; Column : Integer) : Boolean;</p> <pre> Var X : Integer; Y : Integer; ScanResult : Boolean; Begin ScanResult := False; For X := Column - 1 To Column + 1 Do Begin For Y := Row - 1 To Row + 1 Do Begin If (X < 0) Or (X > 9) Or (Y < 0) Or (Y > 9) Then Begin //off the board, so do nothing End Else Begin If (Board[Row][Column] = 'h') And (Board[Y][X] <> 'h') And (Board[Y][X] <> '-') Then ScanResult := True; End; End; End; End; End; End; RadarScan := ScanResult; End;</pre>

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
11	<p><u>MakePlayerMove</u></p> <p>Procedure MakePlayerMove(Var Board : TBoard; Var Ships : TShips);</p> <p>Var</p> <p>Row : Integer;</p> <p>Column : Integer;</p> <p>ShipType : String;</p> <p>Begin</p> <p>...</p> <p>ShipType := '';</p> <p>Case (Board[Row][Column]) Of</p> <p>'A': ShipType := 'Aircraft Carrier';</p> <p>'B': ShipType := 'Battleship';</p> <p>'D': ShipType := 'Destroyer';</p> <p>'S': ShipType := 'Submarine';</p> <p>'P': ShipType := 'Patrol Boat';</p> <p>'F': ShipType := 'Frigate';</p> <p>End;</p> <p>WriteLn('Hit ', ShipType, ' at (', Column, ',', Row, ').');</p> <p>Board[Row][Column] := 'h';</p> <p>End;</p> <p>End;</p>



INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
12	<p><u>domShips</u></p> <pre> ... Begin Row := Random(10); Column := Random(10); HorV := Random(3); If HorV = 0 Then Orientation := 'v'; Else if HorV = 1 Then Orientation := 'd'; Else Orientation := 'h'; Valid := ValidateBoatPosition(Board, Ship, Row, Column, Orientation); End; Writeln('Computer placing the ', Ship.Name); PlaceShip(Board, Ship, Row, Column, Orientation); End; End; </pre> <p><u>ValidateBoatPosition</u></p> <pre> ... Begin Valid := True; If ((Orientation = 'v') Or (Orientation = 'd')) And (Row + Ship.Size > 10) Then Valid := False Else If ((Orientation = 'h') Or (Orientation = 'd')) And (Column + Ship.Size > 10) Then Valid := False Else Begin If Orientation = 'v' Then Begin For Scan := 0 To Ship.Size - 1 Do If Board[Row + Scan][Column] <> '-' Then Valid := False; End; End Else Begin For Scan := 0 To Ship.Size - 1 Do If Board[Row][Column + Scan] <> '-' Then Valid := False; End; End; End; End; End; End; End; </pre>

INSPECTION COPY

COPYRIGHT
PROTECTED



```

Else If Orientation = 'h' Then
Begin
  For Scan := 0 To Ship.Size - 1 Do
    If Board[Row][Column + Scan] <> '-' Then
      Valid := False;
    End // ; removed here
  End
Else
Begin
  For Scan := 0 To Ship.Size - 1 Do
    If Board[Row][Column + Scan] <> '-' Then
      Valid := False;
    End;
  End;
  ValidateBoatPosition := Valid;
End;

```

PlaceShip

```

...
If Orientation = 'v' Then
  For Scan := 0 To Ship.Size - 1 Do
    Board[Row + scan][Column] := Ship.Name[1]
  End
Else If Orientation = 'h' Then
  For Scan := 0 To Ship.Size - 1 Do
    Board[Row][Column + Scan] := Ship.Name[1] // ; removed here
  End
Else
  For Scan := 0 To Ship.Size - 1 Do
    Board[Row + Scan][Column + Scan] := Ship.Name[1];
  End;
End;

```

```

C:\Question12.exe
Computer placing the
Computer placing the
The board looks like
  1  2  3  4
  S  S  S  S
  1  2  3  4
  2  3  4  5
  3  4  5  6
  4  5  6  7
  5 B  D  D  D
  6  B  B  B
  7  B  B  B
  8  B  B  B
  9  B  B  B
  10 B  B  B
MAIN MENU
1. Start new game
2. Load training game
3. Load saved game
4. Board Test
9. Quit
Please enter your cho

```

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
13	<p><u>MakePlayerMove</u></p> <pre> Procedure MakePlayerMove(Var Board : TBoard; Var Ships : TShips); Var Row : Integer; Column : Integer; ShipNum : Integer; X : Integer; Begin GetRowColumn(Row, Column); If (Board[Row][Column] = 'm') Or (Board[Row][Column] = 'h') Then Writeln('Sorry, you have already shot at the square (', Column, ', ', Row, '). Please try again'); Else If Board[Row][Column] = '-' Then Begin Writeln('Sorry, (', Column, ', ', Row, ') is a miss.'); Board[Row][Column] := 'm'; End Else Begin ShipNum := -1; For X := 0 To 5 Do // 5 is the number of ships If Board[Row][Column] = Ships[X].Name[1] Then Begin ShipNum := X; Ships[ShipNum].size := Ships[ShipNum].size - 1; End; Writeln('Hit ', Ships[ShipNum].Name, ' at (', Column, ', ', Row, ').'); Writeln('There are ', Ships[ShipNum].size, ' pieces of ', Ships[ShipNum].Name); If Ships[ShipNum].size = 0 Then Writeln('YOU SUNK MY ', Uppercase(Ships[ShipNum].Name)); Board[Row][Column] := 'h'; End; End; End; End; </pre>

```

CAQuestion13.pas
The board looks like:
0 1 2
0 0 0
1 0 0
2 0 0
3 0 0
4 0 0
5 0 0
6 0 0
7 0 0
8 0 0
9 0 0
Please enter column:
Please enter row:
Hit Destroyer
There are 0 pieces of Destroyer
YOU SUNK MY DESTROYER
You have 17 turns left
Do you want to continue?

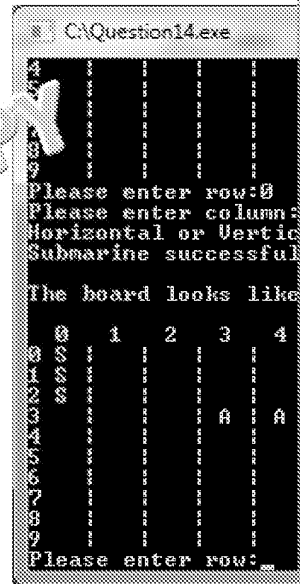
```

INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
14	<p>DisplayMenu</p> <p>Writeln('5. Manually place ships');</p> <p>Main program</p> <p>Else If MenuOption = 5 Then</p> <p>Begin</p> <p>PlaceManualShips(Board, Ships);</p> <p>PlayGame(Board, Ships);</p> <p>End</p> <p>PlaceManualShips</p> <pre> procedure PlaceManualShips(Board:TBoard; Ships:TShips); var Count : Integer; Row : Integer; Column : Integer; Valid : Boolean; Orientation : Char; begin Orientation := ' '; Row := 0; Column := 0; For Count := 0 To 5 Do // number of ships begin Valid := False; While Not Valid Do begin Writeln('Please enter row:'); Readln(Row); Writeln('Please enter column:'); Readln(Column); Writeln('Horizontal or Vertical (h/v)'); Readln(Orientation); Valid := Valid or PlaceShip(Board, Ships[Count], Row, Column, Orientation); If Not Valid then Writeln('Invalid location. Please choose again.'); <pre> Readln; Writeln(Ships[Count].Name, ' successfully placed at (' , Row, ', ', Column, ')'); PlaceShip(Board, Ships[Count], Row, Column, Orientation); RealBoard(Board); end; end; end;</pre> </pre>



INSPECTION COPY

COPYRIGHT
PROTECTED



Question	Answer
15	<p><u>Score</u></p> <pre> Type TScore = Record Name : String; Score : Integer; End; TScores = Array[0..4] Of TScore; <u>SetUpScores</u> Procedure SetUpScores(var Scores:TScores); begin Scores[0].Name := 'George'; Scores[0].Score := 17; Scores[1].Name := 'Paul'; Scores[1].Score := 19; Scores[2].Name := 'John'; Scores[2].Score := 23; Scores[3].Name := 'Ringo'; Scores[3].Score := 25; Scores[4].Name := 'Bryan'; Scores[4].Score := 35; end; <u>DisplayMenu</u> WriteLn('6. Display hi-score table'); <u>DisplayHS</u> Procedure DisplayHS(scores:TScores); Var Index : Integer; begin for Index := 0 To 4 Do WriteLn(Scores[Index].Name, ' ', Scores[Index].Score); end; end; </pre>

INSPECTION COPY

COPYRIGHT
PROTECTED



Main program

```
Else If MenuOption = 6 Then  
    DisplayHS(scores)
```

Declarations near start

```
Var
```

```
    Board : TBoard;  
    MenuOption : Integer;  
    Ships : TShips;  
    Scores : TScores;  
    Sure : Char;  
    Name : String;
```

Program

```
begin
```

```
    SetUpScores(Scores);
```

```
    MenuOption := 0;
```

```
    While (MenuOption <> 9) Do
```

```
    Begin
```

```
        SetUpBoard(Board);
```

```
        SetUpShips(Ships);
```

```
        DisplayMenu;
```

```
        MenuOption := GetMainMenuChoice();
```

```
        If MenuOption = 1 Then
```

```
        Begin
```

```
            PlaceRandomShips(Board, Ships);
```

```
            PlayGame(Board, Ships, Scores);
```

```
        End
```

```
        Else If MenuOption = 2 Then
```

```
        Begin
```

```
            LoadGame(TrainingGame, Board);
```

```
            PlayGame(Board, Ships, Scores);
```

```
        End
```

```
        Else If MenuOption = 3 Then
```

```
        Begin
```

```
            Writeln("Enter filename:");
```

```
            Readln(FileName);
```

```
            LoadGame(FileName, Board);
```

```
            PlayGame(Board, Ships, Scores);
```



INSPECTION COPY

COPYRIGHT
PROTECTED



```

End
Else If MenuOption = 4 Then
Begin
    PlaceRandomShips(Board, Ships);
    RealBoard(Board);
End
Else If MenuOption = 5 Then
Begin
    PlaceManualShips(Board, Ships);
    PlayGame(Board, Ships, Scores, Scores);
End

```



```

Procedure PlayGame(Board : TBoard; Ships : Tships; Var Scores: TScores);

```

```

Var

```

```

    GameWon : Boolean;

```

```

    Torpedoes:Integer;

```

```

    Yn : Char;

```

```

    Filename : String;

```

```

Score : Integer;

```

```

Begin

```

```

    Torpedoes := 20;

```

```

    GameWon := False;

```

```

Score := 0;

```

```

    While Not(GameWon) And (Torpedoes > 0) Do

```

```

    Begin

```

```

        PrintBoard(Board);

```

```

        MakePlayerMove(Board, Ships);

```

```

Score := Score + 1;

```

```

        Torpedoes := Torpedoes - 1;

```

```

        Writeln('You have ', Torpedoes, ' torpedoes left');

```

```

        GameWon := CheckWin(Board);

```

```

    End

```



```

    Begin

```

```

        Writeln('All ships sunk!');

```

```

        Writeln;

```

```

If Score < Scores[4].Score Then

```

```

    Begin

```

INSPECTION COPY

COPYRIGHT
PROTECTED




```

    Scores[4].Score := Score;
    Writeln('Well done, you got a hi score');
    Write('Enter your name: ');
    Readln(Scores[4].Name);
    BubSortScores(Scores);
End;
End

```

BubSortScores

```

Procedure BubSortScores (var Scores: Scores);
Var
    Flag : Boolean;
    Temp : Score;
    Count : Integer;
Begin
    Flag := True;
    While Flag Do
    Begin
        Flag := False;
        For Count:=0 To High(Scores) - 1 Do
            If Scores[Count].Score > Scores[Count+1].Score Then
            Begin
                Temp := Scores[Count];
                Scores[Count] := Scores[Count+1];
                Scores[Count+1] := Temp;
                Flag := True;
            End;
        End;
    End;
End;

```

INSPECTION COPY

COPYRIGHT
PROTECTED



Ideas for modifications	How to

INSPECTION COPY

COPYRIGHT
PROTECTED



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:

INSPECTION COPY

COPYRIGHT
PROTECTED



Programming Theory Question

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)	

INSPECTION COPY

COPYRIGHT
PROTECTED



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	

INSPECTION COPY

COPYRIGHT
PROTECTED

