Question 1

*This question is for validating the menu option is between 1 and 9 – this also includes a* ***Try and Catch*** *to ensure that it validates if an invalid symbol is entered by testing your ability to write* ***iteration*** *and* ***selection*** *statements.*

01 This question refers to the subroutine GetMainMenuChoice

The **Skeleton Program** currently has no validation for the menu choice. It should **not** be possible to choose an option that is outside the range 1 to 9 inclusively. The user also should **not** be able to enter anything other than an Integer value for the menu choice.

If the user enters an invalid menu option the program must display the message:

“Incorrect menu option – please try again”

01.1 Include your amended PROGRAM SOURCE CODE for the subroutine GetMainMenuChoice (6)

01.2 SCREEN CAPTURE(S) for a test showing the following values: (4)

0

A

10

1

Question 2

*This question is one way to validate the co-ordinates to ensure that they lie within the range of 0-9 inclusively. This question tests your ability to use* ***selection*** *and* ***iteration****. See question 8 for an alternative way of validation that includes creating your own Function (If this question seems straight forward – complete Question 8 instead)*

02 This question refers to the subroutine GetRowColumn

The **Skeleton Program** currently has no validation for the co-ordinates. It should **not** be possible to choose an option that is outside the range 0 to 9 inclusively for both row and column.

If the user enters an invalid co-ordinate the program will ask for the co-ordinate to be entered again.

“Invalid Co-ordinate – please re-enter”

02.1 Include your amended PROGRAM SOURCE CODE for the subroutine GetRowColumn (4)

02.2 SCREEN CAPTURE(S) for a test showing the following values: (3)

Column: -1

Column: 10

Column: 0

Row: -1

Row: 10

Row: 9

Question 3

*This question shows that you can write your own* ***procedure (no parameters),*** *call this procedure and alter the menu that has been provided. This tests your ability to use* ***procedures****,* ***procedure calls****,* ***selection*** *and* ***output.***

03 This question refers to the subroutine DisplayMenu

Currently there are just three options:

1. Start new game

2. Load training game

9. Quit

Add a further four options:

3. Save a game

4. Set-Up Ships Manually

5. Display High-Score Table

6. Display Instructions

Write a new subroutine called DisplayInstructions. This subroutine should display the following information to the user:

The aim of the game is to destroy all the ships that have been hidden in a 10 x 10 grid called the board.

<press enter to continue>

Alter the subroutine called Main to ensure that DisplayInstructions is called with the correct menu option chosen.

03.1 Include your amended PROGRAM SOURCE CODE for the subroutine DisplayMenu (2)

03.2 Include your amended PROGRAM SOURCE CODE for the subroutine DisplayInstructions (5)

03.3 Include your amended PROGRAM SOURCE CODE for the subroutine Main (3)

03.4 SCREEN CAPTURE(S) for a test showing the menu and option 6 chosen (3)

Question 4

*This question allows you to test your ability at adapting the code to* ***pass a new parameter****. This ensures that you can make use correctly of parameter passing in order to track the number of shots. This question will also test your ability to be able to edit existing* ***selection*** *routines and* ***iteration*** *routines to end the game accordingly.*

04 This question refers to the subroutine PlayGame and MakePlayerMove

At the moment, the player can continue to play the game until the board is full. The program should keep track of the number of hits that the player has made and display these, therefore the player can try to beat their personal best.

Study the subroutine PlayGame

Add a variable with the identifier NumberOfShots – this will have to be sent to the MakePlayerMove

Study the subroutine MakePlayerMove

The variable NumberOfShots should be brought in as a parameter and incremented when a correct shot has been fired (e.g. one that causes a hit or a miss – not one that overwrites a square already shot at).

The routine will also display a message:

You have made X shots so far.

Alter the subroutine PlayGame so that if 30 moves are made the game is over – a message should be displayed “No more shots left – Game Over”

04.1 Include your amended PROGRAM SOURCE CODE for the subroutine PlayGame (5)

04.2 Include your amended PROGRAM SOURCE CODE for the subroutine MakePlayerMove (5)

04.3 SCREEN CAPTURE(S) for a test showing four moves being made: (4)

7,7

6,7

7,7

1,1

04.4 SCREEN CAPTURE(S) for a test showing 30 moves being made and the message “No more shots” (1)

Question 5

*This question tests your ability to edit an existing* ***Data Structure*** *by adding a new data item. You will be required to alter existing code in order for your new ship to be incorporated within the game.*

Q5

This question refers to the subroutine *SetUpShips* which is used to set-up each of the ships. Currently there are five different classes of ships:

Aircraft Carrier

Battleship

Submarine

Destroyer

Patrol Boat

A new category of ship is to be added:

Cruiser of length 4

You will also need to edit the size of the array **TShip** in the subroutine *Main().*

1. Include your amended PROGRAM SOURCE CODE for the subroutine SetUpShips
2. Include your amended PROGRAM SOURCE CODE for the declaration of Ships in subroutine Main
3. SCREEN CAPTURE(S) for a test being made:

Test – by starting a new game and the Cruiser should display on the screen (not hidden)

In order to include the Cruiser in the game you will need to amend the code in the following routines:

Subroutine CheckWin and Subroutine PrintBoard

1. Include your amended PROGRAM SOURCE CODE for the subroutine CheckWin
2. Include your amended PROGRAM SOURCE CODE for the subroutine PrintBoard

Question 6

*This question tests your ability to use the letter assigned to the array to look up the name of the ship in the data structure and display this if the ship is hit. This question tests your ability to make use of the* ***array****, the* ***data structure for ships*** *and also refer to* ***Strings as an array of characters****.*

Q6

This question refers to the subroutine *MakePlayerMove* which is used to work out if a move has been a miss or a hit. The player it currently told “Hit at (Column,Row)” but it would be better if it said which enemy ship they had hit.

Modify the code so that it also outputs the name of the ship if there has been a hit.

HINT:

The Board will hold the first letter of the ship that has been hit before it is replaced with a “h”

Use a For Loop to cycle through each ships to match the letter from the board with the first letter of each ship

If they match – then output the full ship name.

Your message should look like this:

“Hit Aircraft Carrier at (6,6)”

1. Include your amended PROGRAM SOURCE CODE for the subroutine MakePlayerMove

Test your code by loading the training map, then entering

a column value of 8 and a row value of 1

a column value of 7 and a row value of 6

a column value of 3 and a row value of 1

1. SCREEN CAPTURE(S) for the tests being made:

Question 7

*This question extends on Question 6. This question tests your ability to add an a****dditional variable to the Data Structure*** *Ships and then alter the* **data items** *to incorporate these changes. This question will allow a sunken ship to receive a message – You sunk my battleship.*

Q7

Following on from the last question – when a Ship is sunk – the program should display the message:

“You sunk my BattleShip (Name of Ship)” and indicate which ship has been sunk.

To complete this task you will first need to add to the existing structure:

1. Add a variable to the Structure TShip called Hits – this will be used to keep track of the number of hits that the ship has had.

Include your amended PROGRAM SOURCE CODE for the Structure titles TShip

1. Study the procedure MakePlayerMove – if a hit has occurred then you should perform the following:

* Identify which ship has been hit
* Add one to the Hits for that particular ship
* If the number of Hits is the same as the size of the ship then output the message

“You sunk my Battleship – Name”

Include your amended PROGRAM SOURCE CODE for the subroutine MakePlayerMove

Test your code by loading the training map, then entering

a column value of 1 and a row value of 4

a column value of 5 and a row value of 9

a column value of 1 and a row value of 5

a column value of 6 and a row value of 9

a column value of 4 and a row value of 9

a column value of 7 and a row value of 9

1. SCREEN CAPTURE(S) for the tests being made:

Question 8

*This question is one way to validate the co-ordinates to ensure that they lie within the range of 0-9 inclusively. This question tests your ability to use* ***Functions****,* ***selection*** *and* ***iteration****. See question 2 for an alternative way of validation that doesn’t include creating a Function (If this question seems too difficult – complete Question 2 first to become confident with validation and then come back to Question 8)*

08 This question refers to the subroutine GetRowColumn

The **Skeleton Program** currently has no validation for the co-ordinates. It should **not** be possible to choose an option that is outside the range 0 to 9 inclusively for both row and column.

If the user enters an invalid co-ordinate the program will ask for the co-ordinate to be entered again.

“Invalid Co-ordinate – please re-enter”

Due to Row and Column both taking a value between 0 and 9 – it would be inefficient to have a validation loop for each – therefore a Function will be used to check the validity of the data entered.

08.1

Create a new Function called ***ValidateCoOrd*** that takes a coordinate as a parameter

The Function will check that the co-ordinate lies within the range of 0 to 9 inclusively and if it doesn’t it will output the error message:

“Invalid Co-ordinate – please re-enter”

If the Co-ordinate is valid, the function will return True, if the coordinate is invalid it will return False.

Include your amended PROGRAM SOURCE CODE for the subroutine ValidateCoOrd (4)

08.1

Study the procedure GetRowColumn

After the user has input Column – call the Function ValidateCoOrd to check that the co-ordinate entered is valid. If False is returned, the user must be repeatedly asked to enter the co-ordinate.

After the user has input the Row – make use of the Function ValidateCoOrd a second time.

Include your amended PROGRAM SOURCE CODE for the subroutine GetRowColumn (4)

08.2 SCREEN CAPTURE(S) for a test showing the following values: (3)

Column: -1

Column: 10

Column: 0

Row: -1

Row: 10

Row: 9

Question 9

*This question relates to the ability to save a game so that it can be resumed later. This question makes use of* ***Procedures****,* ***Parameters****,* ***FileWriter****,* ***Arrays****,* ***Iteration****.*

09 Currently the program does not allow games to be saved.

Write a procedure called **SaveGame** that takes the file name and the board as parameters.

The procedure will save the same to the file name specified

09.1

Include your amended PROGRAM SOURCE CODE for the subroutine SaveGame (9)

09.2

Study the procedure PlayGame

After the board has been displayed the user should be prompted with the following message:

“Do you wish to save the game (Y for Yes) or any other key to continue?”

If the user selects “Y” then the user is asked to enter a filename for the game and the game is saved making use of the SaveGame procedure.

If the user selects any other key the game continues.

Include your amended PROGRAM SOURCE CODE for the subroutine PlayGame (6)

09.2 SCREEN CAPTURE(S) showing three moves made and the game saved as saveGame1.txt

Show a SCREEN CAPTURE of the contents of the file saveGame1.txt (5)

09.3 Study the subroutine DisplayMenu

Add a menu option:

7. Load Game

Study the subroutine Main

If the user selects option 7 – to load the game then the user is prompted for the name of the file and the game is loaded.

Include your amended PROGRAM SOURCE CODE for the subroutine DisplayMenu (1)

Include your amended PROGRAM SOURCE CODE for the subroutine Main() (4)

Question 10

*This question relates to the ability to use the Data Structure (Array and Ships) and alter the way in which a ship is placed (e.g. Diagonal). This question makes use of* ***Arrays****,* ***Structures****,* ***Selection****,* ***Iteration****.*

10 Currently the program does not place ships diagonally – this routine is going to allow placing of ships in diagonal from top to bottom, left to right

Study the subroutine PlaceRandomShips:

Alter the procedure so that a random number is generated between 0 and 2 and that if the random number is 0, orientation is set to v, if the random number is 1, orientation is set to h and if the random number is 2, orientation is set to d.

Study the subroutine ValidateBoatPosition

Add an option to the first part of the If Statement to check that the ship size would not go off the board if diagonal is chosen

Add a third part to the second selection statement to check that each next place in the diagonal is currently a blank.

Study the PlaceShip routine:

Add a third option to the If statement that places the ship diagonally (top to bottom, left to right)

10.1

Include your amended PROGRAM SOURCE CODE for the subroutine PlaceRandomShips (3)

10.2

Include your amended PROGRAM SOURCE CODE for the subroutine ValidateBoatPosition (6)

10.3

Include your amended PROGRAM SOURCE CODE for the subroutine PlaceShip (3)