**DOCUMENTING THE CODE(1)**

1. Read the Preliminary Material.
2. Identify aspects of the program that can be improved or where it breaks.
3. Identify a list of parameters for each function and procedure and state whether they are passed by reference or value.

**CHALLENGES(1)**

1. **Procedure DisplayMenu** - Modify the code to display a welcome message to the User at the start, include the title of the game.
2. Add an option so that the game displays a set of instructions to the User. (Keep the instructions brief).
3. Modify the menu and the main program to allow a User to SAVE a game (only provide the option).
4. Modify the menu and the main program to allow the User to LOAD a previous SAVED game (Not the training game and only provide the option).
5. **Main Program** – alter the program ( and add a function) that asks the User for their name.
6. **Procedure PlaceRandomShips** – Alter the program so that it displays the number of spaces each warship takes on the board.

**CHALLENGES(2)**

1. **Procedure PlayGame** - Modify your code to display how many valid guesses the User has taken (excluding repeat guesses)
2. Display the total number of guesses as a score at the end of the game as well as the User’s name.
3. **Function GetMainMenuChoice** - Validate the User input from the menu to only allow those options included in the menu.
4. **Procedure GetRowColumn** – alter this procedure to validate the row and column input so that it is restricted to the grid only. (0..9,0..9).

**CHALLENGES(3)**

1. **Hierarchy** – Draw out a hierarchy chart for this program. This will show how each procedure/function is called. The top of the hierarchy is the main program.
2. **Function GetMainMenuChoice** - The menu choice currently allows letters as well as numbers, with the program returning an error if a character is entered. Use Try…..except to further validate the User input by asking the User to only type in a number and at the same time managing this exception. The following code is an example. (note: press continue when an error occurs.)

1. **Main program** – define a SET as a global variable that includes the numbers 0..9. Use a subset of this (1,2,9) in the main program and if the User choice is not in this subset then output a message to the User. (You will need to alter this set to include those options that you have added). Example
2. **Procedure GetRowColumn** – At this point you have changed the program so that only 0..9 for both row and column are valid. Use Try..except to further validate so that a message is outputted if characters are entered by the User.

**CHALLENGES(4)**

1. CONST – define a constant in the program to define the number of ships. Make this constant equal to seven. Use this constant in the appropriate functions and procedures in the program. This will mean that the number of ships in the game can change with only having to update the constant before running the program.
2. **CONST** – set the board dimensions as constants and use them in your code. Be careful what name you call each dimension.
3. **Procedure PrintBoard** – Modify the code (comment out the unnecessary line) so that the location of the ships are printed each time the board is displayed. This will help while you alter other aspects of the game. You will be able to see where the ships are placed.
4. **Procedure SetUpShips** – Modify this code to add another ship, frigate of size two.
5. Using a FOR loop add in a total of THREE frigates. This should then give you a total of eight ships in the game.
6. **Procedure PrintBoard** – Alter this code to allow for the presence of a Frigate.
7. **Function CheckWin** – similar to challenge 23

**CHALLENGES(5)**

1. **Number of turns** – Set a global variable called NoOfTurns. Write your own function or procedure that calculates (see below) and sets the number of turns the User can have in a game. This should be an integer. You will need to determine how and when your procedure/function is called.

Number of turns = total length of all ships x 2

1. Output a message to the User to show how many turns they have at the start and how many they have left after each turn. Repeats do not count.
2. Test this procedure by altering the length of the patrol boat to 3. Check that the number of turns has changed.
3. Alter your function/procedure so that the multiplier can be changed by the User based upon the level of difficulty. Let the User choose the level of difficulty. (The User can enter their level in the MAIN program.

Easy, multiplier = 4

Intermediate, multiplier = 3

Hard, multiplier = 2

1. **Procedure PlayGame** – Modify this procedure to check whether the number of turns is equal to zero and if it is then end the game. The main menu should then re-appear.
2. **Procedure PlayGame** – Modify this procedure to calculate the User score once they have used all their turns or if they win the game. The scoring system is one point for every hit (EASY LEVEL) AND one point for any unused turns, if there are any.
3. Check that above function/procedure so that the User is given two points at Intermediate level and three points at Difficult level for each hit but still only one point for each unused turn.

**CHALLENGES(6)**

1. Write a procedure to SAVE the current players name and their score to a file. Call this procedure at the end of a game through a menu option and append the data to the file. Set the name of the file as a constant.
2. Modify the program so that the User can QUIT in the middle of a game.
3. Write a procedure to SAVE the current game. The User should be able to save the game as a menu option. Set the name of the file as a constant. You will have to consider what data needs to saved.
4. Write a procedure to LOAD a previously saved game. The User should be able to load this game as a menu option.

**CHALLENGES(7)**

1. Write a procedure that opens a file that loads the records of the ships. This file should be loaded as an alternative to the procedure SetUpShips. (You will need to initially create the file.)
2. Load the file that you have used to save the scores and the player’s names. SORT this data using descending score. Output only the top 4 scores and the name of the player for each score.
3. The User name has to be of length 3. Write the code to validate this.
4. Validate the User difficulty level when entered by the User.

**CHALLENGES(8)**

1. Convert your game into a TWO player game.
2. Inform the User when a ship has been destroyed.
3. When a ship is destroyed, then replace the letter ‘h’ with the letter ‘x’.
4. Show the position of the ships at the end of the game if the player hasn’t won.
5. Use DATE functions to calculate how long a User has played a game.
6. Display horizontal lines as part of the board.
7. Alter the record type TShip, adding a Boolean variable called ‘sunk’. Set this variable to FALSE for each ship.