**UNIT 16 ASSIGNMENT 1**

**Task 1 – P3**

**Purpose of program:**

The purpose of the program will be for the user to input the depth, length and width of their pond, it will then work out an area based on the given figures. After the area is figured out, depending on whether it fits into gauge 1 or gauge 2 the price will be worked out, being area \* £1.12 for gauge 1 and \* £1.76 for gauge 2. The program will only work out these values if the maximum length is 10 meters with a maximum width of 10 meters and a maximum depth of 2 meters. The purpose will be to figure out the price of the liner depending on the size of the pond.

**Console screen design:**

Please enter the pond length in meters:

Please enter the pond width in meters:

Please enter the pond depth in meters:

The area of liner needed is:

This will cost £\_\_\_\_\_

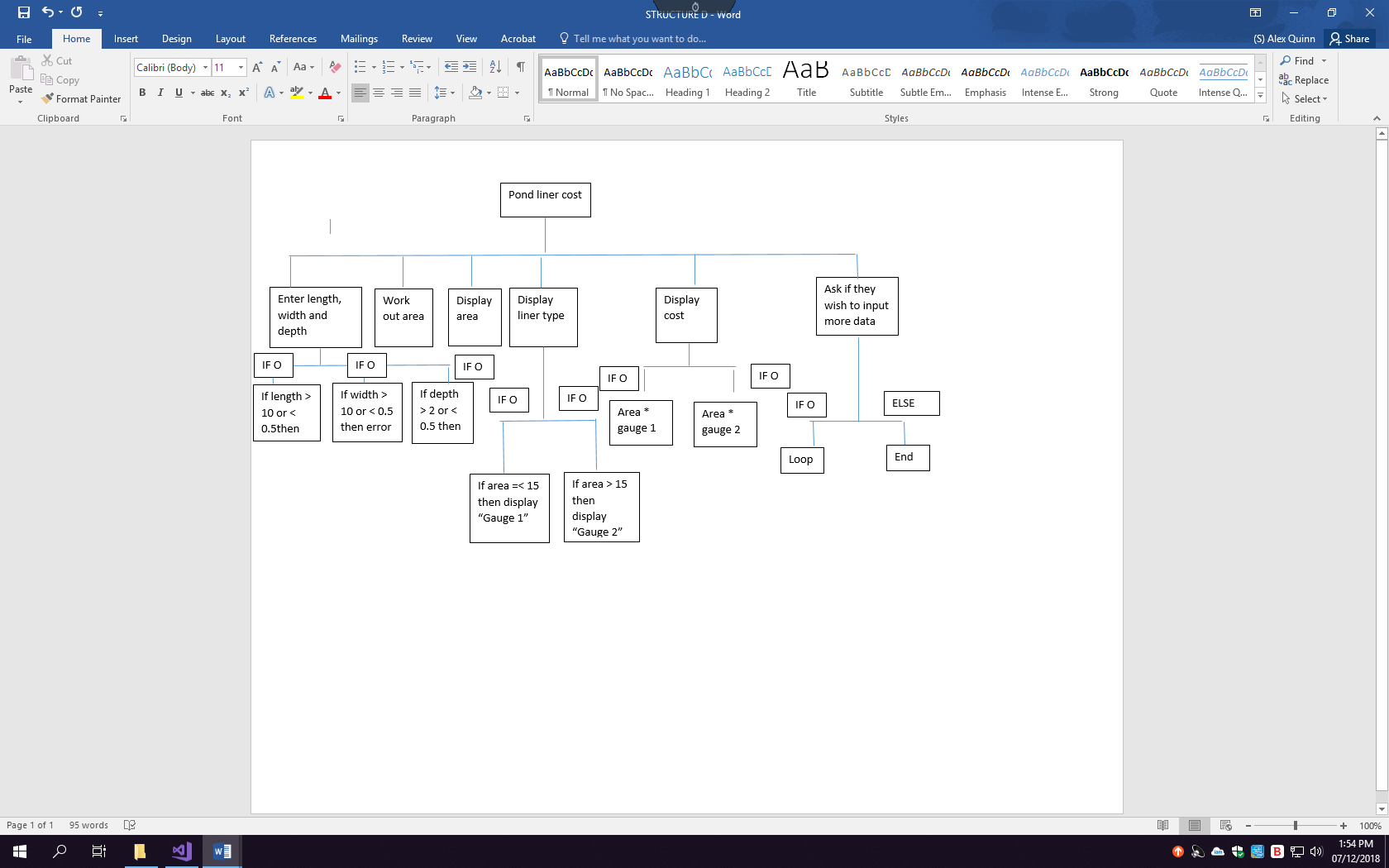
The type of liner needed is:

Do you wish to continue? Please enter ‘Yes’ or ‘No’

**Data Dictionary:**

|  |  |  |
| --- | --- | --- |
| Variable | Data type | Purpose |
| Length | Single | This can hold any number despite whether it is a whole number or a decimal in regards to the length of the pond. |
| Width | Single | This can hold the width due to the fact that it may be a decimal or a whole number. |
| Depth | Single | This figure may be a decimal number too, meaning it would suit it best. |
| Area | Single | This number will be a multiplication of all three figures, meaning it may be a decimal number again. |
| Gauge 1 | Constant | This figure is fixed at a set value, being £1.12. |
| Gauge 2 | Constant | This would be a set value too being £1.76. |
| Price | Decimal | This would also be a decimal due to the cost being the needed gauge \* the area of the pond. |
| Errorflag | Boolean | This will be used to determine whether the user input is correct or not. |
| Choice1 | String | This will be used within the error flag, and will determine the users input. |
| Choice1uppercase | String | This will be the same as the choice1 function but will be used with the UCase() function, to validate the errorflag data. |

**Structure Diagram**



**Pseudocode for the program:**

Start

Get input for length

Get input for width

Get input for depth

Display area of liner needed

Display type of liner needed (=< 15 then gauge 1 else gauge 2)

Display cost (Area of pond \* necessary gauge)

Get input for loop

End

**CODE: Task 2 – P4**

'All of my variables are set outside of the routines to make them global variables, meaning that they can be used in all routines without being redefined.

Dim length As Single

Dim width As Single

Dim depth As Single

Dim area As Integer

Const gauge1 = 1.12

Const gauge2 = 1.76

Dim price As Decimal

Dim errorflag As Boolean

Dim choice1 As String

Dim choice1uppercase As String

Sub Main()

'The DO initialising a loop and an error flag to validate the input data.

Do

'The error flag will ensure that the program loops until the correct data has been input in order for all the subroutines to be ran successfully, if the flag is false, the code will run.

errorflag = False

'This routine is used to calculate the area of the pond, by obtaining the length, width and depth.

Areacalc()

'if the flag is false then the entire code will run throughout.

If errorflag = False Then

'This sub routine is used to define which gauge of liner is needed for the calculation.

typeliner()

'This sub routine is used to calculate the overall price of the liner for the pond.

priceL()

Else

'This string works with the error flag to alert the user that the figures input are invalid.

Console.WriteLine("ERROR IN INPUT")

End If

Console.WriteLine("Do you wish to continue? Please enter 'Yes' or 'No' "

'The uppercase function allows for whatever input to be valid with the loop, by converting all inputs to uppercase.

choice1 = UCase(Console.ReadLine())

choice1uppercase = UCase(choice1)

'Once the choice matches the necessary conditions, the loop will end and discontinue the code.

Loop Until choice1 = choice1uppercase

End Sub

Sub Areacalc()

Console.WriteLine("Please enter the pond length in meters: ")

length = Console.ReadLine()

'This if statement makes sure that the figure given is within range for the liner, if it isn't an error message will display.

If length > 10 Or length < 0.5 Then errorflag = True

Console.WriteLine("Please enter the pond width in meters : ")

'Setting the variable to the read line, allows for the users input to be allocated. E.g. if 4 is input, width will equal 4.

width = Console.ReadLine()

'This statement also makes sure that the figure is within the necessary range for the calculation to work.

If width > 10 Or width < 0.5 Then errorflag = True

Console.WriteLine("please enter the pond depth in meters: ")

depth = Console.ReadLine()

'Again, if the measurement given is not correct then the message will show the user that it is incorrect.

If depth > 2 Or depth < 0.5 Then errorflag = True

'This sum calculates the area of liner needed, following the given calculation.

area = length + width + (depth \* 4)

Console.WriteLine("The area of liner needed is " & area)

Console.ReadLine()

End Sub

Sub typeliner()

'This statement defines which gauge of liner is needed, which will then be displayed to the user

If area <= 15 Then

Console.WriteLine("The type of liner needed is gauge 1 liner.")

Console.ReadLine()

Else

Console.WriteLine("The type of liner needed is gauge 2 liner.")

Console.Readline()

End If

'The end sub part allows for this routine to be closed off and for the next routine to be ran.

End Sub

Sub priceL()

'This statement calculates the price of the liner depending on which gauge is used, with anything less than or equal to 15 being gauge one.

If area <= 15 Then

price = area \* 1.12

ElseIf area > 15 Then

price = area \* 1.76

End If

'Finally, the price for the liner is displayed to the user.

Console.WriteLine("This will cost £" & price)

Console.ReadLine()

End Sub

**TASK 3 – P5**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Date of test | Data input | Expected result | Actual result | PASS / FAIL | Improvement (if necessary) |
| Test to see that the length can be input | 03/12/18 | 5 | For the length to be input as 5 and for the next command to be asked |  | PASS |  |
| To see whether the width can be input | 03/12/18 | 5 | For the width to be input as 5 and for the next command to be asked |  | PASS |  |
| To see whether the depth can be input | 03/12/18 | 2 | For the depth to be set as 2 and for the next step to take place |  | PASS |  |
| To see whether the area will display before the liner type | 03/12/18 | 5,5,2 | For the area to display then the type and cost |  | FAIL | To fix this I changed the order of the routines. |
| To see whether the area is calculated | 03/12/18 | Length, width and depth (5,5,2) | For the area to be calculated and displayed as 18 |  | PASS |  |
| To see whether the gauge needed will display | 03/12/18 | Length, width and depth (5,5,2) | For gauge 2 to display because this calculation would be -18 |  | FAIL | To fix this I set the gauges to a new variable and sub routine |
| To see whether gauge 1 will display when the area is under 15 | 03/12/18 | Length, width and depth (2,2,2) | For gauge 1 to display as the liner type as this area would be 12 |  | PASS |  |
| To see whether gauge 2 will display when the area is over 15 | 03/12/18 | Length width and depth (10,10,2) |  |  | PASS |  |
| To see whether the price will display | 03/12/18 | Length, width and depth (2,2,2) | For the price to display |  | PASS |  |
| To test whether extreme and boundary data can be entered | 04/12/18 | Length as 10, Width as 10 and depth as 2 | For the calculations to correctly work and display as 28 |  | PASS |  |
| To test whether the bottom extreme data can be input, being 0.5,0.5 and 0.5 | 04/12/18 | All of the inputs as 0.5 | For the calculation to display correctly |  | PASS |  |
| To test whether unexpected data will be flagged as incorrect | 04/12/18 | The input for the depth will be 50, this is outside the boundary | For the console to display “ERROR IN INPUT” |  | PASS |  |
| For the loop option to be displayed when after the price is shown | 04/12/18 | 5,5,2 for the lengths and “yes” to continue | For the question to appear asking whether the user wishes to continue |  | PASS |  |
| To test whether the loop will function and allow the program to restart without closing | 04/12/18 | “yes” will be input in response to the question. | For the loop to run, asking the first question again. |  | PASS |  |
| To check the figures are accurate when calculated | 04/12/18 | 5,5,2 As the inputs then compared to a calculator with the same formula | For it to display the area as 18 and for the calculator to support this |  | PASS |  |

**DEBUGGING TOOLS:**

**Step through:** This is an action taken within the debugger, where it will “step over” or through a given line, it will then detect whether the line has a function within and then run the function, line by line as the results will be returned without the need for debugging each line. The tool essentially takes the code line by line to highlight or check for errors within.

**Break points:** This allows for the codes execution to be halted or paused at a specific point, where an error has occurred, allowing the editing and debugging of the code by the insertion of the breakpoint. This saves you from having to run the whole code with the error influencing the outcomes, while also allowing you to see where it has occurred within the code, allowing the error to be fixed.

**Compiler messages:** These are messages which displays an error message will display to guide the alteration of the code, giving the reason the error has occurred allowing for the debugging through instruction. These messages occur when there is an error in the compiler itself, therefore meaning that it cannot compile the code. It allows for an easy solution to debugging the source code.

**Conclusion:** I tested the code thoroughly to ensure that all functions, iterations, selections, sequences and inputs worked. By doing this I was able to flag up any errors and was then able to edit the code to cater for this errors. I have tested all aspects of the code as well as typical, extreme and unexpected data, this allowed for assurance that the code is functional in regards to the data input. By testing the selections and iteration it ensured that the correct algorithms were selected to complete the code, e.g. by choosing gauge 1 over gauge 2. To conclude, the code was fully tested to a great extent through being scrutinized step by step to ensure that each line of the code is fully functional in relation to the given sub routine and the whole module to meet and fit the specification completely.

**TASK 5 – P2 AND M2**

|  |  |  |
| --- | --- | --- |
| **CODE** | **Description of tool** | **Reason for use** |
| Dim length As Single  Dim width As Single  Dim depth As Single  Dim area As Integer  Const gauge1 = 1.12  Const gauge2 = 1.76  Dim price As Decimal  Dim errorflag As Boolean  Dim choice1 As String  Dim choice1uppercaseAs String | **I have declared the users input as a single data type, which will then temporarily store the number**  **The area is declared as an integer, which will store it as the necessary number**  **The constants are the gauges for the price because it can keep the value the same throughout**  **The error flag is a Boolean and the choices as strings, which store text as data** | I have used variables to temporarily store data to the appropriate name given, from the users input to then be used further on within the calculation to work out which liner type and cost will be used for the pond in question. While decimal is used for a cost value and the inputs are single because the measurements will be within a specific range. Boolean will determine whether the inputs are correct or invalid with the error flag and the strings will be used to get user feedback when regarding the loop for the code to re run without closing. |
| Areacalc()  typeliner()  priceL()  Sub priceL()  If area <= 15 Then  price = area \* 1.12  ElseIf area > 15 Then  price = area \* 1.76  End If  Console.WriteLine("This will cost £" & price)  Console.ReadLine()  End Sub | **These are my defined sub routines, relational operators, sequences.**  **Sub routines determine the sequence the code is ran in.**  **Rational operators determine whether something is mathematically correct** | These sub routines allow for the code to be broken down into several steps, as an alternative to writing all of the code in one main routine. The order of the routines defines the order of the running of the code while also making the code easy to read. The routines are responsible for different tasks that will be executed in the order given. The sub routines allow for a sequence to be put within the code, allowing for an order of execution when the code is ran to make sure that the correct data has been received before the next step is ran, for example, the sequence allows for the measurements to be received before the price is displayed, because without the measurements the price could not be calculated. |
| Do  errorflag = False  Areacalc()  If errorflag = False Then  typeliner()  priceL()  Else  Console.WriteLine("ERROR IN INPUT")  End If  Console.WriteLine("Do you wish to continue? Please enter 'Yes' or 'No' ")  Console.Readline()  choice1 = UCase(Console.ReadLine())  choice1uppercase = UCase(choice1)  Loop Until choice1 = choice1uppercase | **This is the iteration loop, the iteration allows for the code to be ran multiple times without closure, as long as the correct data is input** | The iteration is used for the loop function which will allow for the user to re run the code without having to close and open the program again, as long as the parameters are met to do so, if the users input is No, then the iteration will not loop but if the users input is Yes, the loop will start the code again from the beginning asking again for the user to input the measurements needed. The code will only close if the user inputs No, otherwise any data will be considered as true, meaning that the response will warrant the code to re start |
| errorflag = False  If errorflag = False Then  If length > 10 Or length < 0.5 Then errorflag = True | **These are my error flag uses, which validates the input data to the range specified.**  **I have also used a logical operator within here being the Boolean, returning as true or false.** | The error flag is used to validate the users input, if the figure given within the beginning of the program does not fit the given range this will prompt the error flag to become true and will then display the error message for the user to alert them that an input was incorrect within one of the numbers given, either being an unexpected value or being a value outside of the valid numbers.  I have used a Boolean here because the result can only be true or false, meaning that the logical operator will determine whether or not the input data is correct. |
| If length > 10 Or length < 0.5 Then errorflag = True  Console.WriteLine("Please enter the pond width in meters : ")  Else  Console.WriteLine("ERROR IN INPUT") | **This is an IF statement used for the length while also using the error flag, to determine which statement is to be displayed depending on which “if” is met – This is known as selection.** | The statements are responsible for displaying the figures input while checking that they are within the specification needed, they will display the error message if the input is not valid within the range, allowing for accuracy and certainty. By using the error flag it allows for the validity of the value to be confirmed or flagged as incorrect, meaning that the error message will then display.  The data input will allow for a “selection” to be made for the next step within the program, in this example the selection will either continue the code or will display the error message. |
| Sub priceL()  If area <= 15 Then  price = area \* 1.12  ElseIf area > 15 Then  price = area \* 1.76  End If  Console.WriteLine("This will cost £" & price)  Console.ReadLine()  End Sub | **This is the use of Arithmetic, used to mathematically work out the area needed using BIDMAS, while selection is also used** | The use of arithmetic in this code allows for the price to be calculated by using the constant values of the gauges, if the area is under 15 then the gauge 1 value will be used within the calculation while if the area is over 15 then the gauge 2 value will be used to calculate and then display the price to the user. Following this arithmetic within the if statement allows for an accurate price to be displayed through making sure that the correct gauge is used for the area of the pond. Selection is used in this too also determine which value is needed for the calculation. |
| choice1uppercase = UCase(choice1) | **This is the function, this specific function allows for the users input to be converted to uppercase automatically.** | I have used a function within the iteration due to the fact that the user may not enter a case sensitive answer, meaning that if they input no instead of NO then the code will still close because the answer will always be converted to uppercase. Allowing for any variation of the required answer to still work with the iterations termination despite not being completely correct in the set type. |

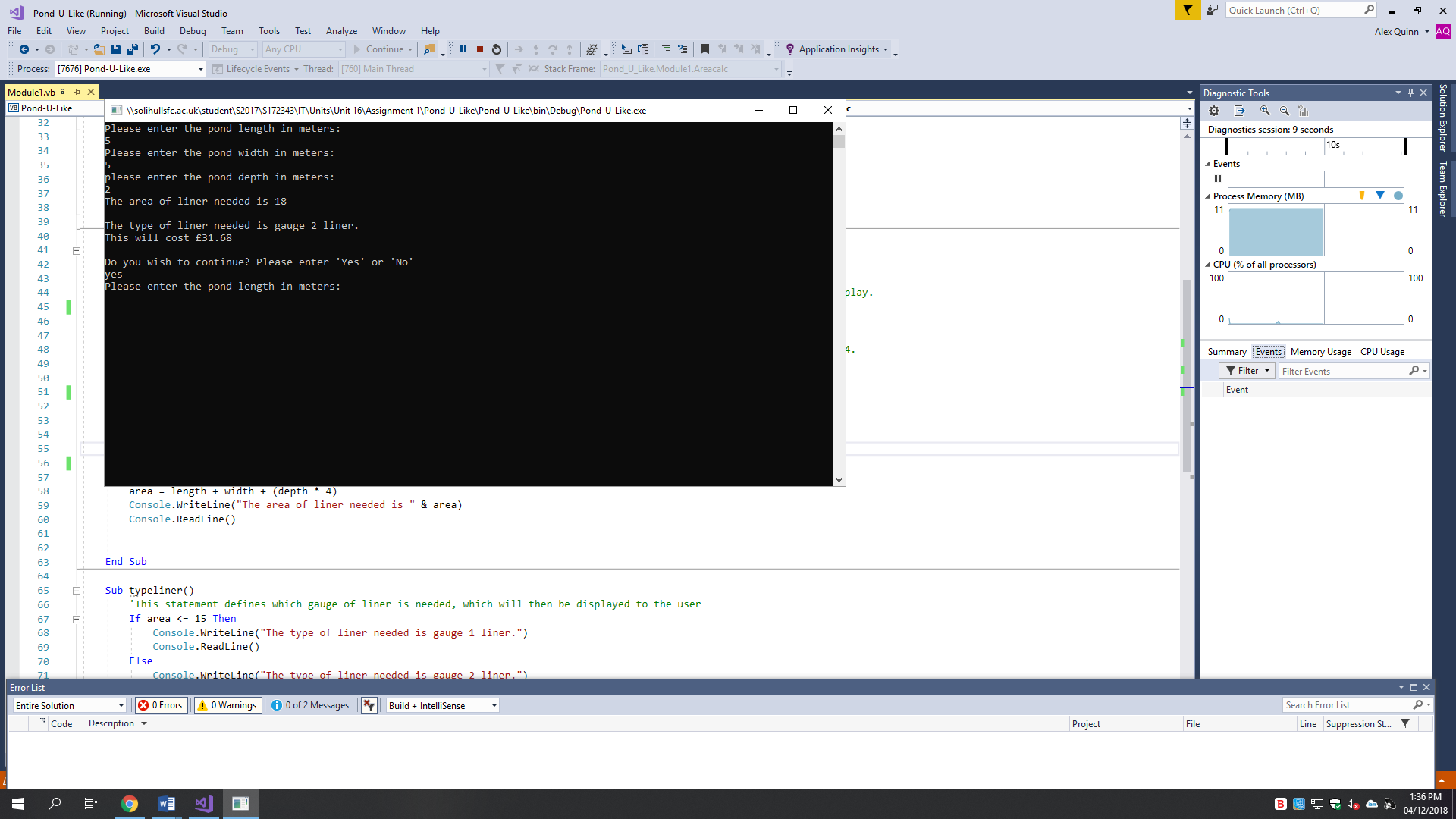
**TASK 6 – D2**

**Purpose of the program:**

Pond-U-Like Ltd constructs environmentally friendly garden ponds using butyl (rubber) liners. They originally carried out all calculations and recorded all details of each pond they constructed on paper, but this system proved too inefficient so they automated part of the process using an Excel spreadsheet.

Here is a brief description of the system used by Pond-U-Like Ltd. They have automated the process of working out the size and gauge (thickness) of liner required for each job (which is based on the size of the pond to be constructed) using a spreadsheet with appropriate formulas. When the chosen site is surveyed, the **dimensions** of the pond are recorded on a data capture form and the necessary calculations are made back at the office where the data is entered on their spreadsheet.

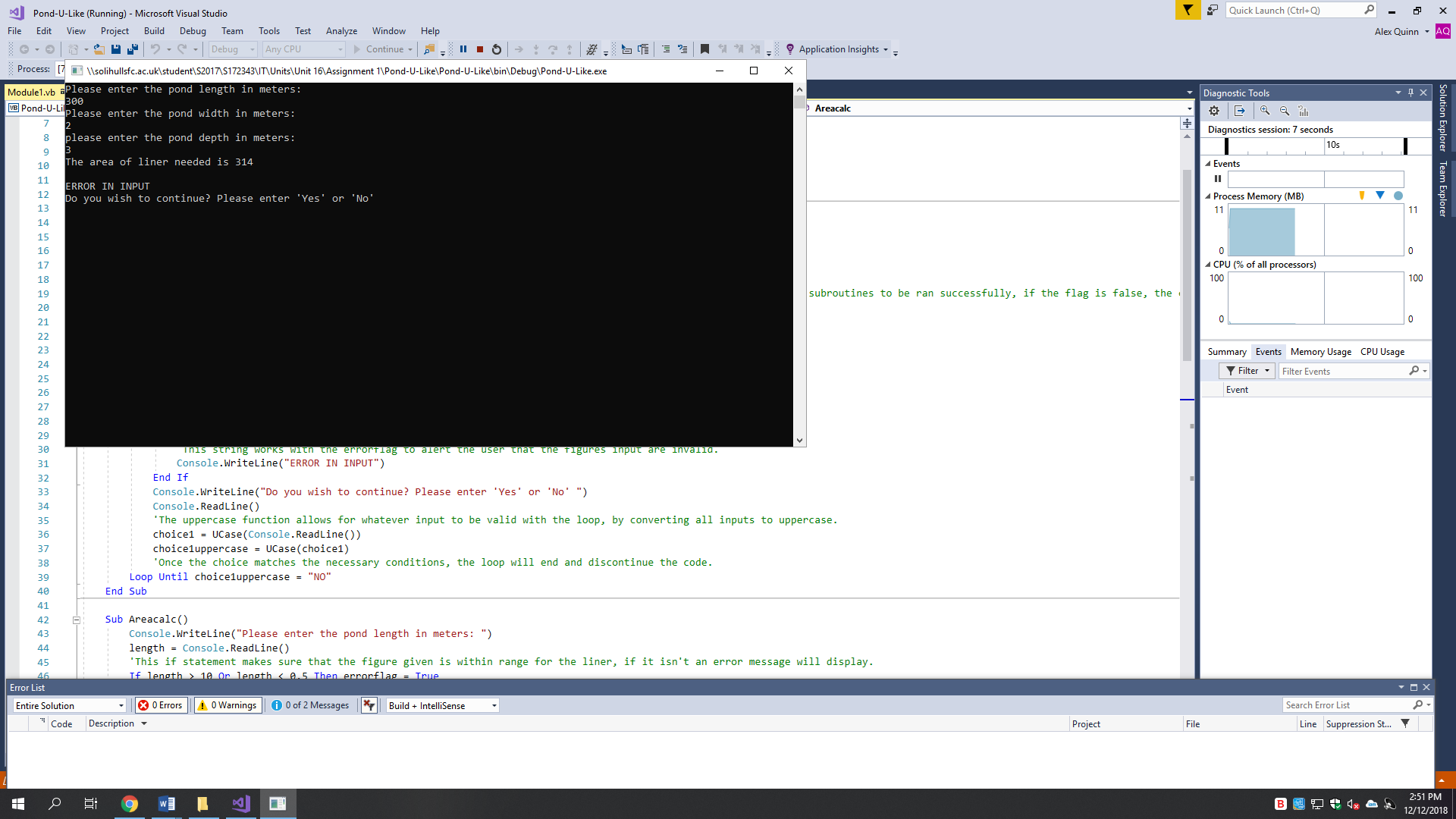
**Degree of success –**

The program is meant to receive the length, width and depth of the pond from the user, it is to then use the measurements to calculate the area with the given formula, length + width + (depth\*4), after this, the code is to multiply the area by the cost of the given gauge. The gauge would be gauge 1 if the area is under 15 and gauge 2 if it is over 15. Using this multiplication, the price and gauge will then be displayed to the user, informing them on how much it will cost. After this code has executed, a loop has been used to allow the user to begin the code from the beginning again. If the data input is incorrect, or outside the maximum and minimum figures allowed the loop will also run, meaning that the area, cost and gauge will only be displayed after the correct measurements are met. The program is not meant to complete if the data input is incorrect, and this does not happen. The code is also not meant to display the wrong gauges or cost, which also does not happen due to the IF statements and arithmetic.

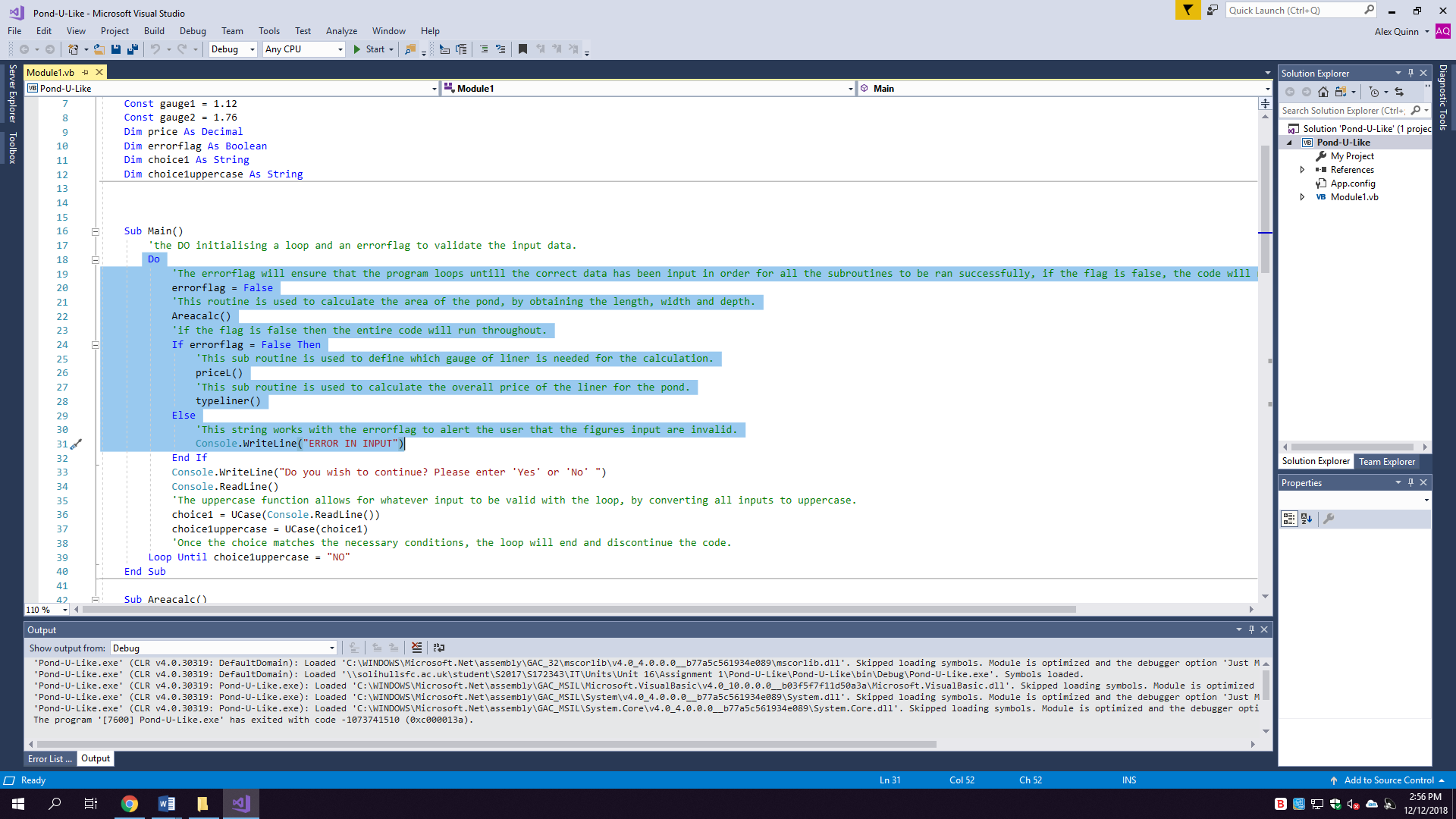
Evidence of these successes can be seen throughout the testing from test 1 to test 15.

**Is there any error handling? –**

I have used an error flag within my code to make sure that the input measurements fit the maximum and minimum lengths. This allows for the code to pick up any errors in the input, for example, if the user was to input a length of 12, an error would display to show the user that there has been an error in the input. There are also error flags for the width and depth to achieve the same goal. The error flag ultimately makes sure that all data used within the code fits the requirements for the calculations to work out the gauge and price needed for the liner. Without the error flag the code would accept any value for the measurements but this wouldn’t suit the requirements due to the fact that Pond-U-Like have stated the maximum and minimum requirements for the liner.



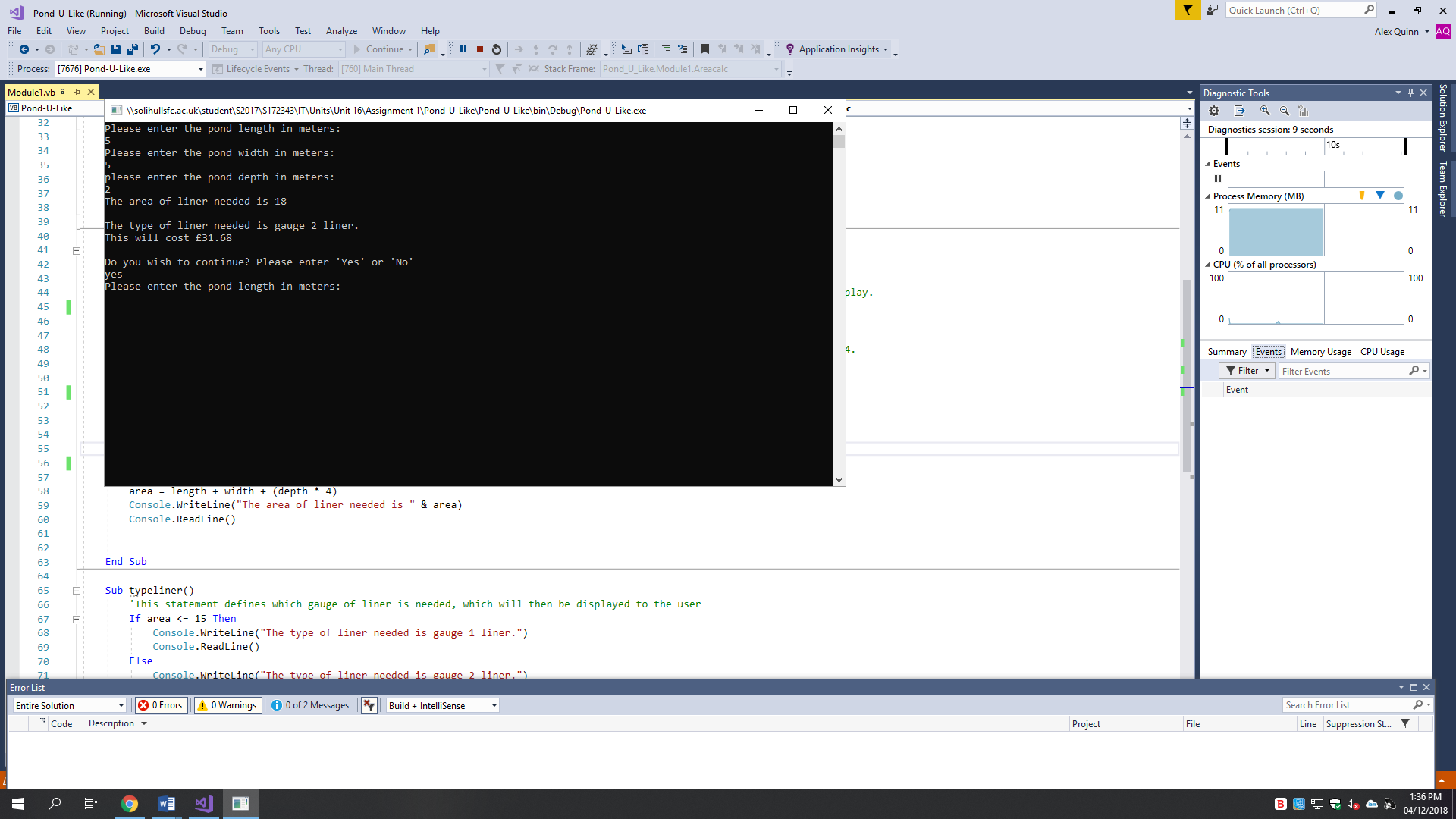
The error flag is used to detect incorrect data, within the code to halt the program by using an iteration to re run the code. These tests can be seen in test 10, 11, 12 and 1



**Is it easy to use? –** The code is easy to use due to the fact that each question is asked individually on separate lines, by doing this is allows it to be easily read line by line while also having correct grammar and an order to the data needed. For example, the syntax allows for each question to have a space in between to display the measurement they have used for the given distance. The program only requires three inputs of data which means that the use is minimal by the user. By displaying the gauge and then the price, it makes it easy to read while also displaying why the cost is what it is. If any errors occur the code alerts the user by displaying the error message and will then ask them if they wish to begin the code over again by inputting one of two options, being yes or no. This allows for any errors to be flagged and corrected by asking the questions over again. The code also always produces the correct result due to the fact that I have used IF statements and a calculation to follow the same needs to meet the specification.

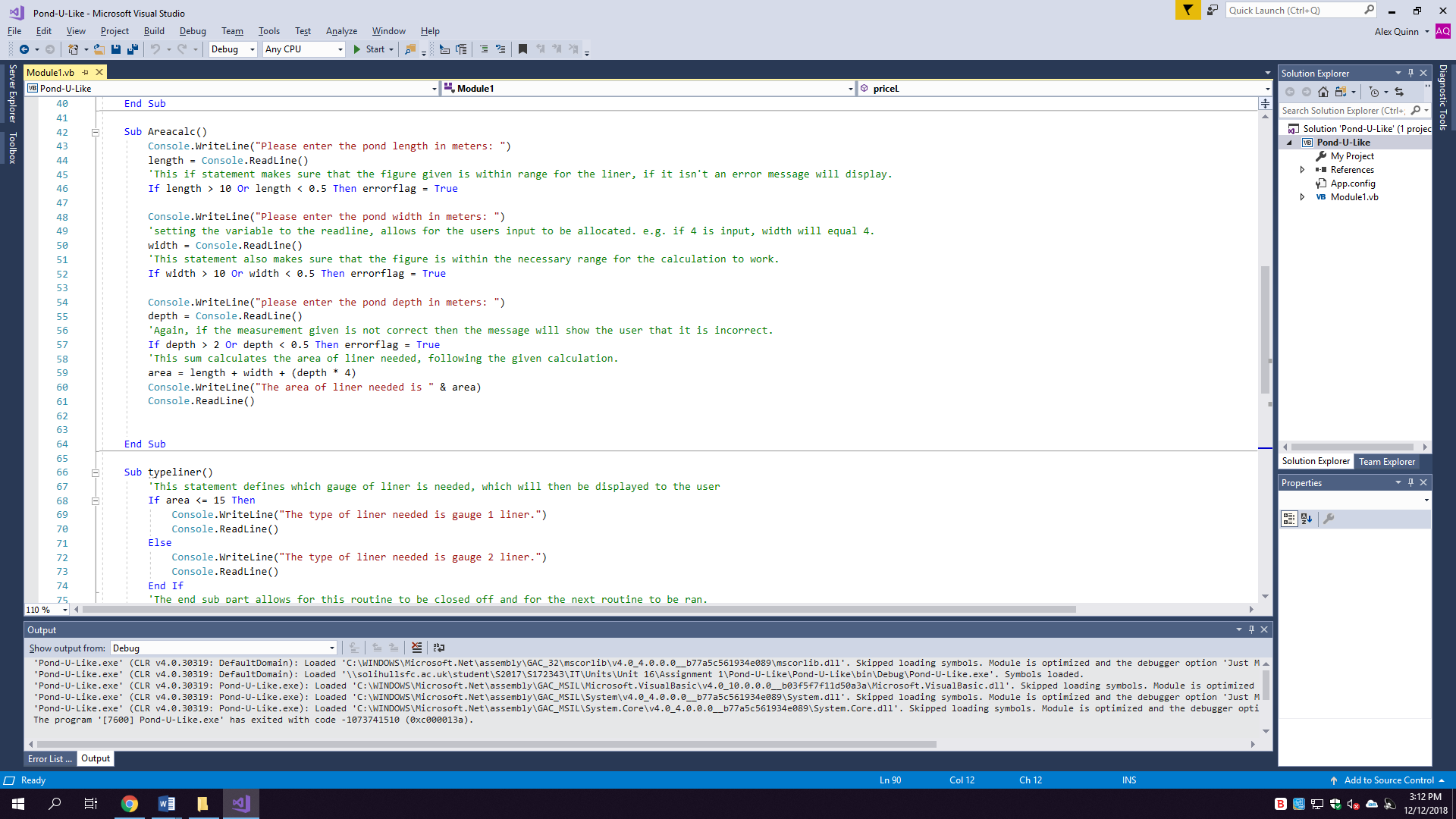
This code shows the arithmetic used to calculate the price and gauge, this can be proved to be successful through tests 5, 7, 8 and 15.

**Are the screen messages suitable? –**

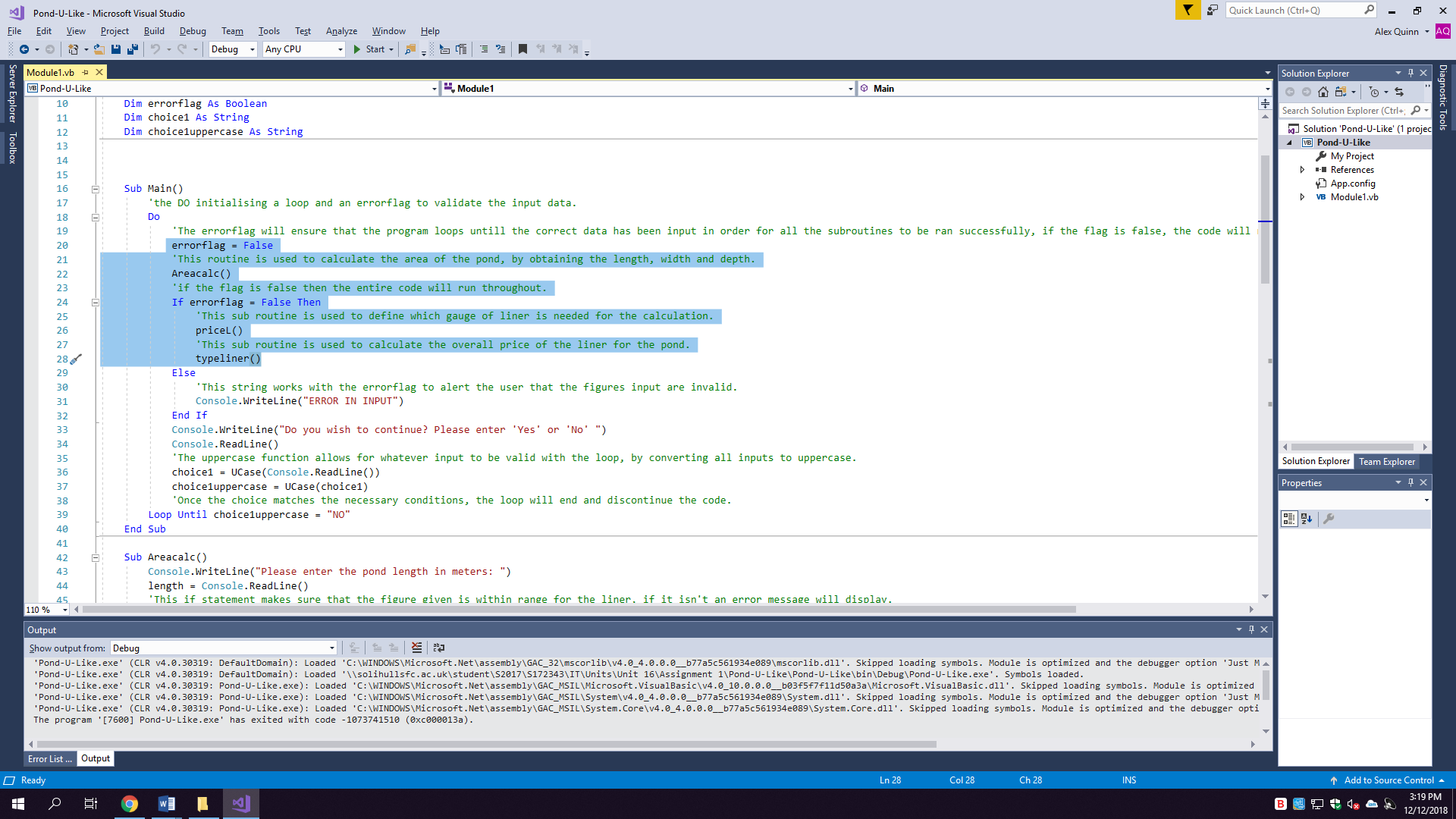
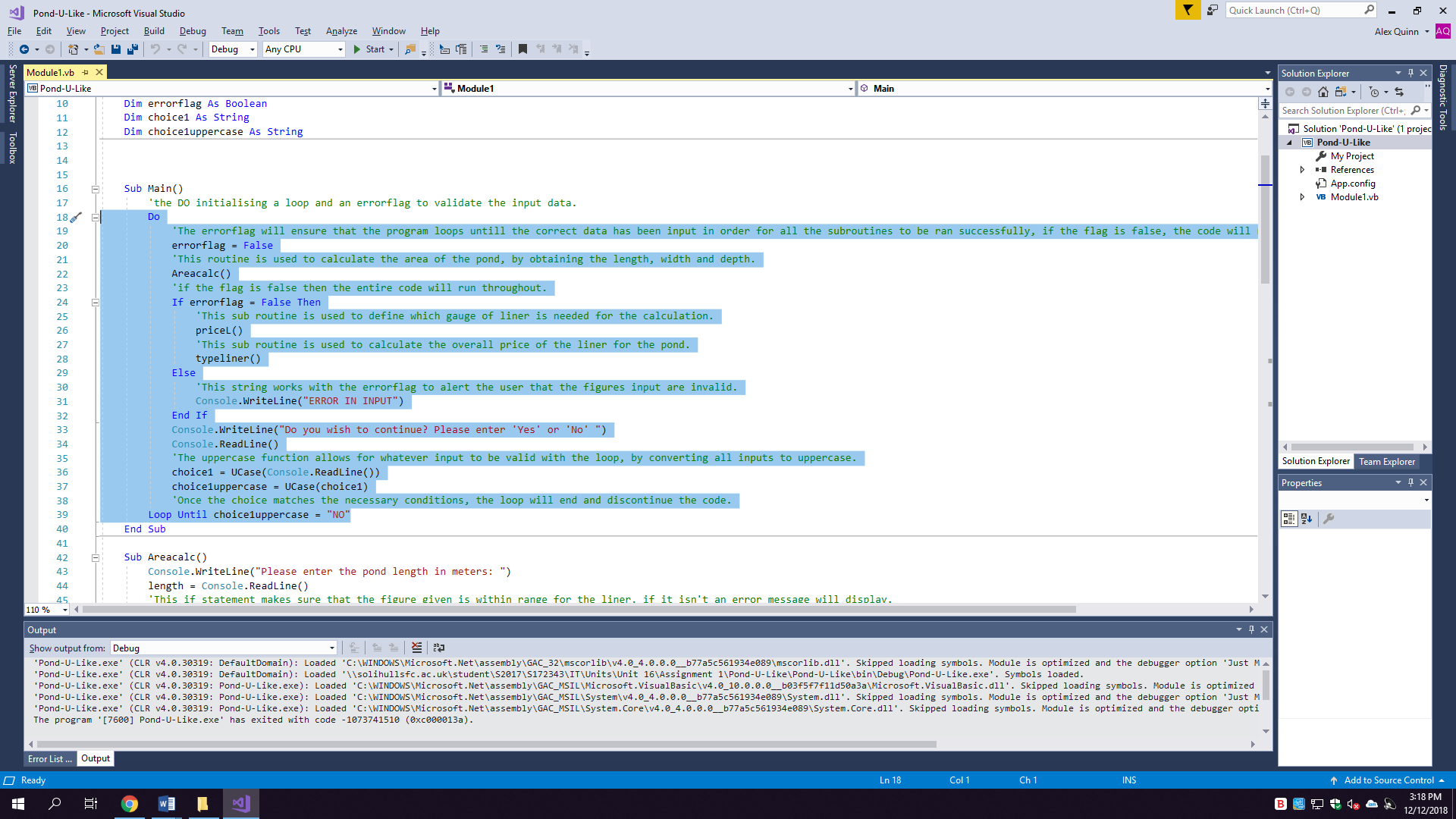
The messages are suitable due to the fact that they display what data is needed and then display what has been done with the data to complete the code. They are suitable due to the fact that they are in relation to what data in required from the user, after all data is received the messages that display the further messages are worded clearly to show the user that the given measurements have been used to calculate the gauge and price given while also having an iteration worded in a way that is clear to warrant a response to execute the next command. If the messages were not suitable then the user may get confused when following the stages within the console, meaning that the incorrect data may be input or the displayed information misread, because of my messages this will not occur.

**The use of iteration, sequence and selection –**

I have used the three programming constructs throughout the program and all 15 tests can support this. The use of these constructs allowed the code to run successfully alongside the specification to achieve the necessary targets. The selection processes (see figure 1) allow for the gauges and price to be calculated while also allowing for the error flag to validate my code when the user inputs and figure, being the measurements, without the selections the code would not be able to display the correct messages or work out the calculations to achieve the criteria set by the scenario. The iteration aspect of my code allows for the executions to repeat several times and also works with the selection of the error flag to decide whether the data is correct to continue (see figure 2). The last construct would be the use of a sequence, I have used a sequence in the main routine of my code by initializing sub routines to break down my code, by doing this it will run the routines in order of the list, which is specifically seen in tests 3, 4 and 6. This allows me to give a structure to the code while also placing it in a logical order for the user. For example, if I was to put the price routine before the area input routine, the code would not work because the user would not be able to input data for the price to be calculated (see figure 3).

Figure 1 -

-

Figure 2 - Figure 3 -

**Possible Improvements:**

**The program could save data –**

The program could be improved to save data for the instance of a returning customer, for example, if a customer was to query the liner on more than one occasion, by saving their data within the program it would allow for the user to skip the program stage of actually inputting data and following each step. By using saved data they could automatically view their previously input data as opposed to having to complete the form again, this would be an improvement for returning customers as it would be more efficient and would allow for multiple accesses to a customer’s file.

**The code could be more explicit as to where errors occur –**

The code could be changed to display the error flag message underneath the actual value of which the error has occurred in, opposed to displaying it after all inputs are displayed and the area is calculated. This would be an improvement as it would give a higher degree of certainty and error awareness as the user may not know where the error has occurred within their measurements. For example, if the user put their length as 13, the error message could display straight away to alert that the length is incorrect or excessive as opposed to displaying it after all inputs are received, by being more explicit with the error and saving the user time when completing the program to receive a price and gauge.

**The program could have a print feature –**

The code could be improved by having a print feature due to the fact that the program and results could then be used as a receipt or as a way to give a customer a quote on a pond liner, this would be an improvement as it would allow for a way to keep physical copies of the client’s data either for Pond-U-Like or for the customer. By having this feature it would mean that a paper copy of the data could be kept. This would be more practical for use when regarding a quote due to the fact that the customer could keep it, as opposed to being told a quote and not directly having a means to keep a record of the gauge, price or data.

**Overall, how well does the program fit the specification?**

The program fully fits the specification although extra improvements could be added to develop the program to a higher capability. The program receives the three values necessary and calculates the necessary values while then displaying all of the stages to the user, allowing them to see how it is all set out. While doing this it also asks for further input to begin the commands again within the loop. By doing these things the program fully meets the specification due to the fact that all criteria are met and that the program is fully functional by completing the steps required to calculate the area, gauge and price. While also being tested and fixed to cater for the errors made within the code, this ensured that the program was to run successfully every time it is ran, fitting the specification completely.