***Group No: 7***

***Introduction***

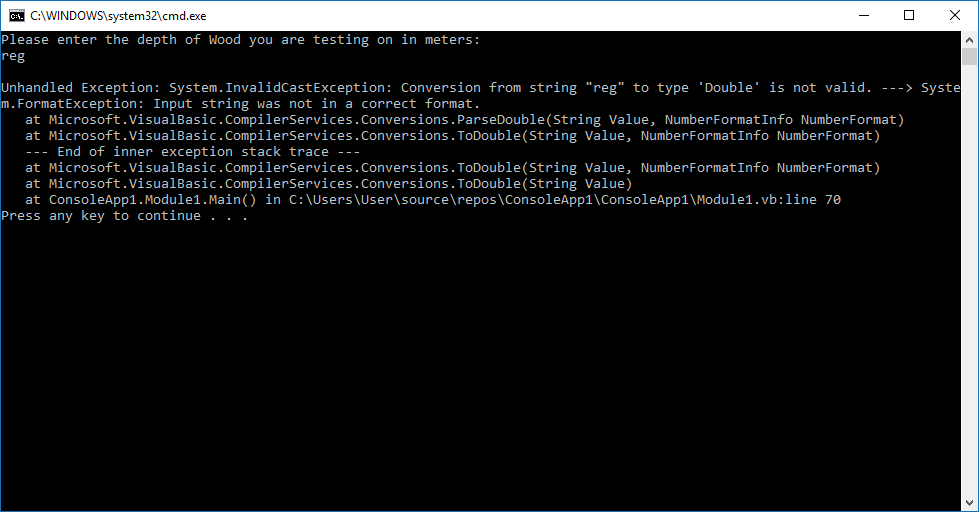
This program is designed to help experimental scientist, asking them to input depths, checking the result (>0) and then going throw a secondary verification process of confirming with them the value they input. The program then prints the period the bullet spent in the medium, and then the average times over the 3 materials the bullet spent in the medium.

***Question 1***

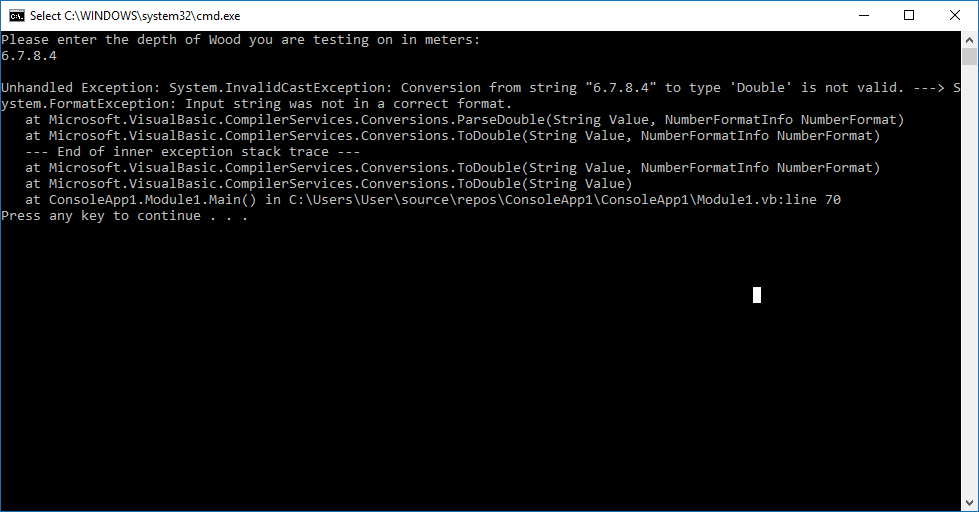
*Introduction: functionality of the program*

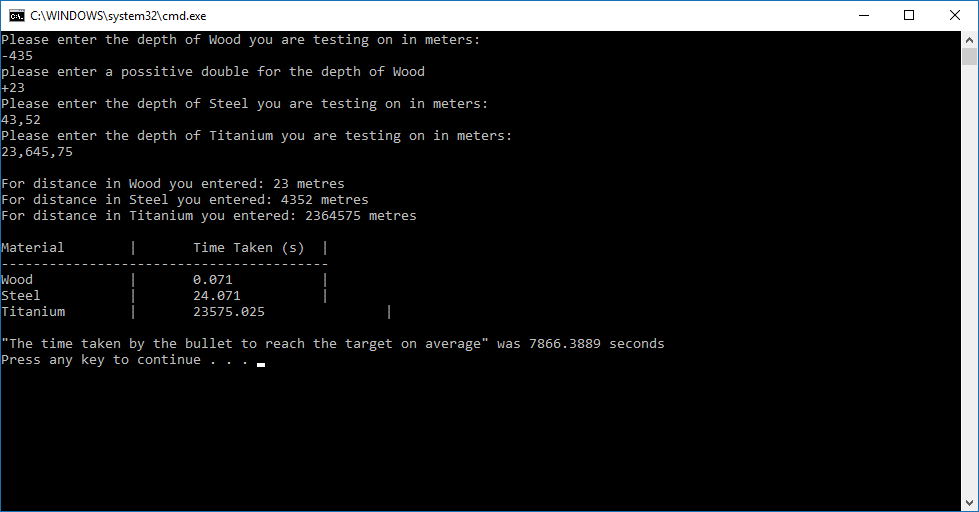
1. *any special instructions or warnings to the user (or assessor!) such as in which case code does not work ,*

* End user should enter a numerical, “.” “,” “-” or “+” otherwise “Unhandled Exception:” will appear and the code will not continue. This means the end user should not input any units after the numerical data; having converted to standard SI units (metres) beforehand. (evidence in bellow screenshot)



* A comer is accepted; however, the user should know it will be removed during the string to double type conversion. They should also be aware “+” and “-“ will affect the calculation.
* The user can not input more than one fall stop otherwise the program will crash(evidence in the second bellow screenshot)





* The user must input a depth greater than 0, other wise they will be asked to input another value (evidence in above screen shot)

1. *draw attention to any aspects of the program that you are particularly proud of (i.e. reusability, maintainability aspects of the code.)*

I’m particularly proud of my nonspecific up to 5 coulomb header generator isolates the heading formatting to one place rather than having to type it our every time. Which got used throw several other questions

I’ve also written functions for every equation which should aid those who don’t intuitively know that t = s/v for instance. Allowing them to manipulate the code in the future which affecting the original equation by accident.

As well as a universal average function

My code is also concise and to the point

*Conclusion:*

In conclusion I have implemented several functions to future adaptation, taking cognitive load off their minds whilst coding. My functions have also been useful to those writing q2 to q4. The functions them selves were written to be as general as possible, thusly being useful in more programs, and being written simply so that they don’t do to much and become more complicated for the user to understand. The code works and fulfils its requirement.

***Question 2:***

*---------------------*

***Question 3:***

*---------------------------*

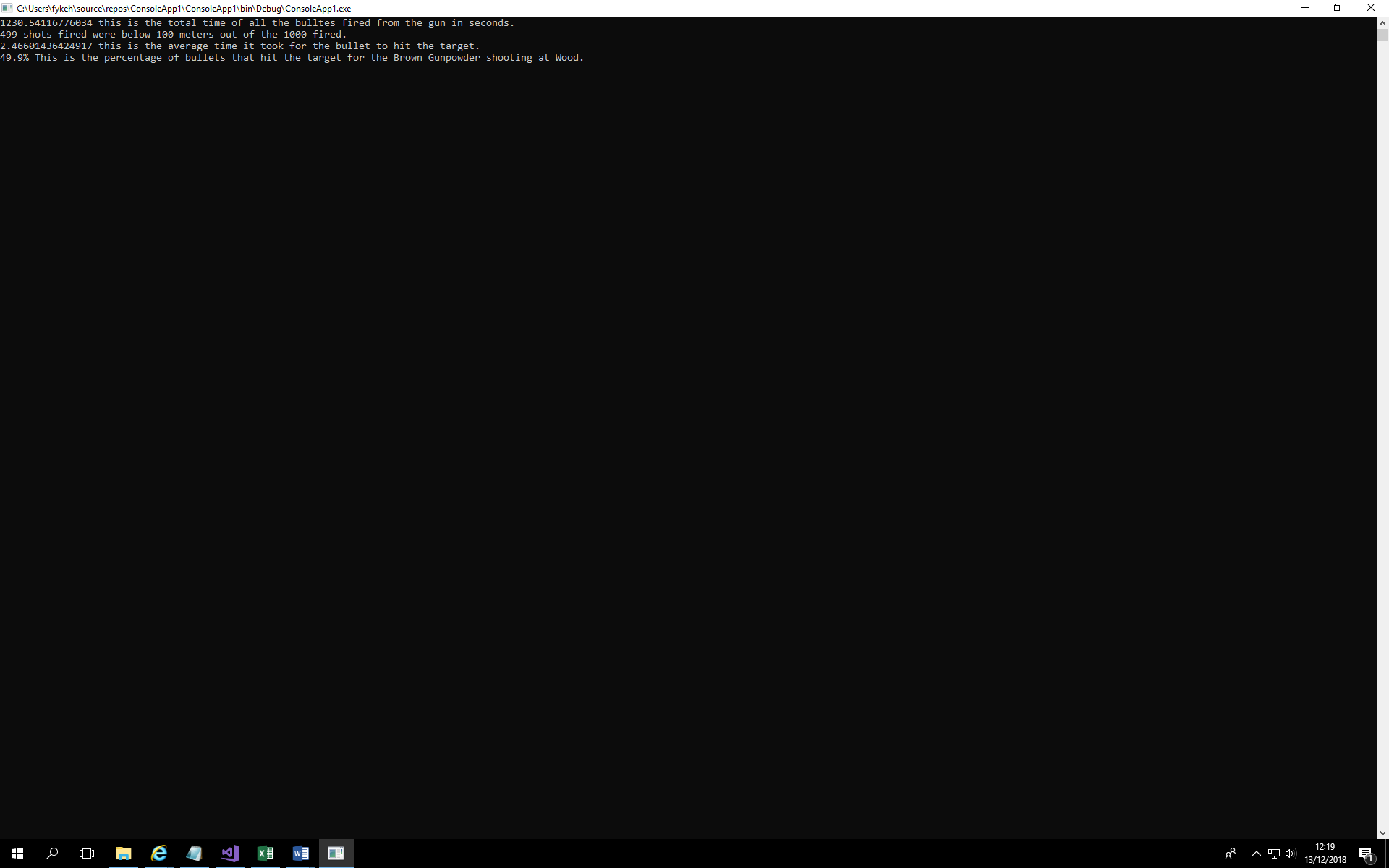
***Question 4:***

1. This programme has been designed to allow the experimental lab to see what effect the different gunpowder’s have on the trajectory of the bullet from the gun, such as the deviation that occurs in its flight path. It also allows them to see the difference between the brown gunpowder and the sulphur-free gunpowder and by analysing the results they can decipher which gunpowder provides the best accuracy.
2. For this code to work the user must enter the following variables:

* **Maximum angle** (degrees, °) of the shot as an integer.
* **Height** (meters, m) of the target as an integer.
* **Distance** (meters, m) that the bullet will be travelling as an integer.
* **Velocity** (meters per second, m/s) the bullet is traveling at which can be inputted as a decimal.
* **Gunpowder** type (brown, sulphur-free)
* **Material** (wood, steel, titanium) the bullet is being shot at.

If any of the above are not entered into the programme it will not run, however the programme will let the user know that they need to be inputted into the code. Also, the values inputted must be entered in the given units for the correct answer to be outputted.

1. I am proud of my code due to its ease of use for the user such as the variables used in this programme have been named so that it is easy for the user to understand what values need inputting in certain parts of the programme. The names of them are clear such as if the height of the target needs inputting the variable is therefore called ‘height\_of\_target’. Furthermore, within the code there are lots of equations that have had to be used to calculate different variables, so next to each equation it states what it is calculating so that if any changes need to be made this allows the user to easily manipulate the programme. Finally, once all the calculations have been made the programme then clearly prints out the results and clearly states what each one is within a sentence.



1. In conclusion this programme allows the users to easily figure out how the different gunpowder types affect the trajectory of the bullet being shot but also can be easily changed to figure out other calculations that may be needed in the experiment. Due to the ease of use of this programme it will allow the users to input the data needed and be given the correct answers back which can reduce their work load which in turn reduces the time taken for the experiment. Also within this question I was able to utilise functions from question 1 and question 2 which will help the user to understand each code due to the same functions being used throughout the programme.

***Overall Conclusion***

*e.g.*

*all/ essential/ core/ a few/ required functions of CW are successfully implemented. There is some initial problems with ….. However later it has been …….*

*use of functions /makes it easy to develop/ cause many problems*

overall all the code fulfils all its requirements for each question, with general functions make it easier for future implementation. We had lots of initial problems which we all had to cover come, some individually and other as groups; for instance, writing to a document, we needed to make it usable across devises and thus a normal file path would not work. Eventually Liam worked out how to create and then write to a file in a local folder within the ConsoleApp so that we could all work on it and the user could run it in Visual Studios without it crashing.

***Word count***