***Group No: ?***

***Introduction***

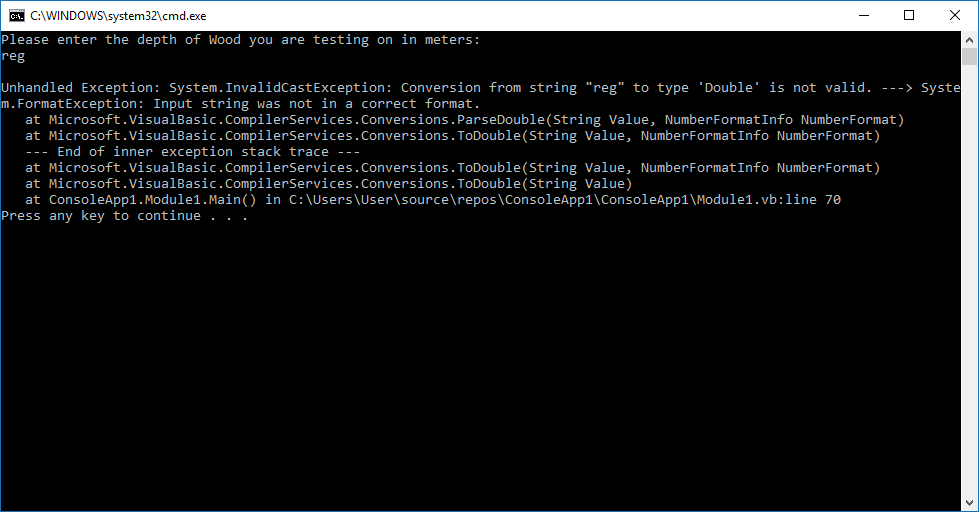
This program is designed to be helpful to the scientists who will be using it, asking them to input depths, checking whether the are valid field results, aka above 0; and then going throw a secondary verification process of confirming with them the value they input. the program will then inform the scientists 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***

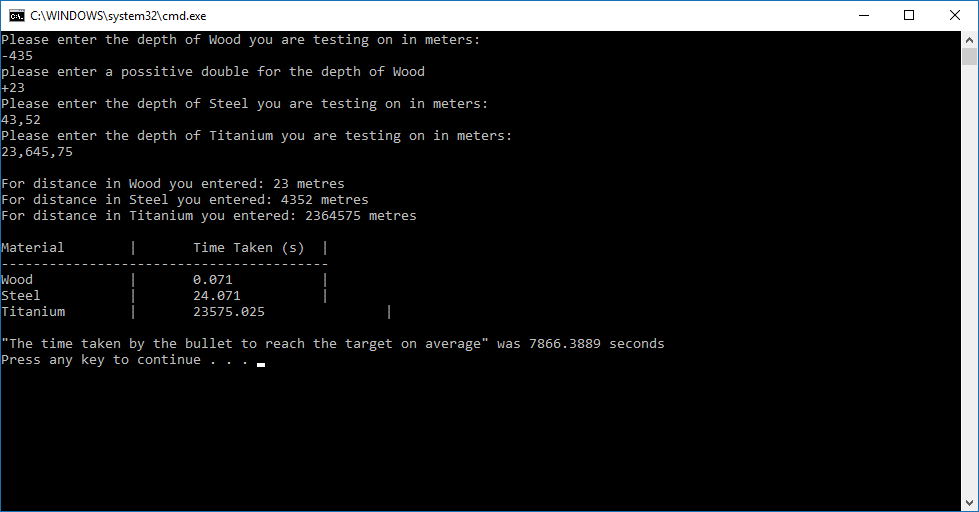
*Introduction: functionality of the program*

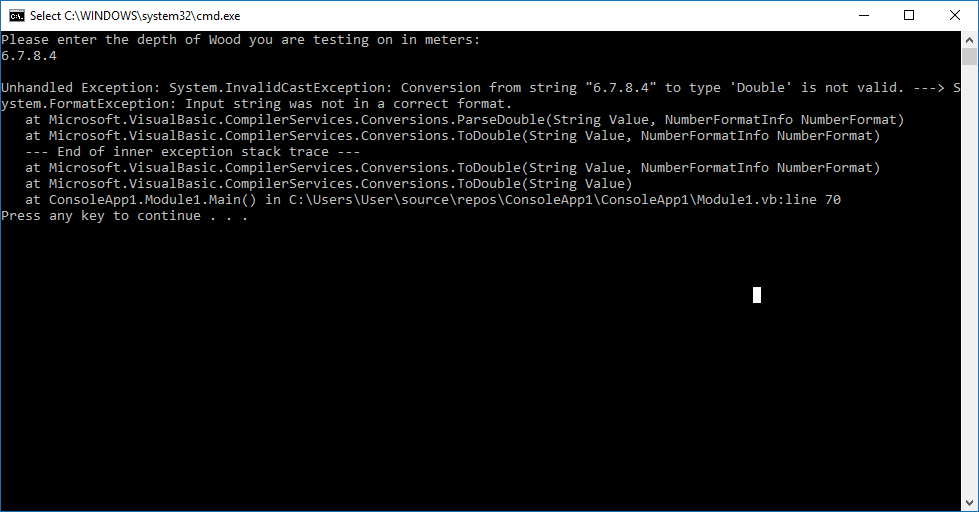
1. *any special instructions or warnings to the user (or assessor!) such as in which case code does not work ,*

* Any value other than a numerical, point or comer and it will output an “Unhandled Exception:”, complaining about not being able to convert the inputted string to type double. 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)



* The only other Characters other than integers which are accepted are comer which but is removed during this string to double type conversion, thusly the calculated for time will take the depth as inputed without the commer (evidence in bellow screenshot). Fallstopes (decimal places) and “-” & “+” can also be inputed however their can be no more than one fallstop in the users input as this will crash because it carnt convert a string with multiple fallstopes into a double (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 which is above 0/a positive double (evidence in above screen shot)

*e.g.* ***Any user input validation***

***e.g. Any limitation of the program***

*e.g.* ***Evidence: Tested by the user input …. and the result is ….***

1. *or to 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. It also seems intuitive to use, how many coulombs do you want and then a list of what you want to put inside the coulomb.

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.

***e.g. use of variables and why , what functions are used and why..***

*Conclusion:*

In conclusion I have implemented several functions to aid future programmers re-write this code as well as writing new programmes, taking cognitive load off their minds whilst coding because possibly long complicated calculations have been graphically simplified to a function name. 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 as keep them generally usfull

*e.g. all/ essential/ core/ a few required functions of the question 1 are successfully implemented.*

*use of functions /are challenging /makes it easy /…..*

***Question 2:***

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

***Question 3:***

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

***Question 4:***

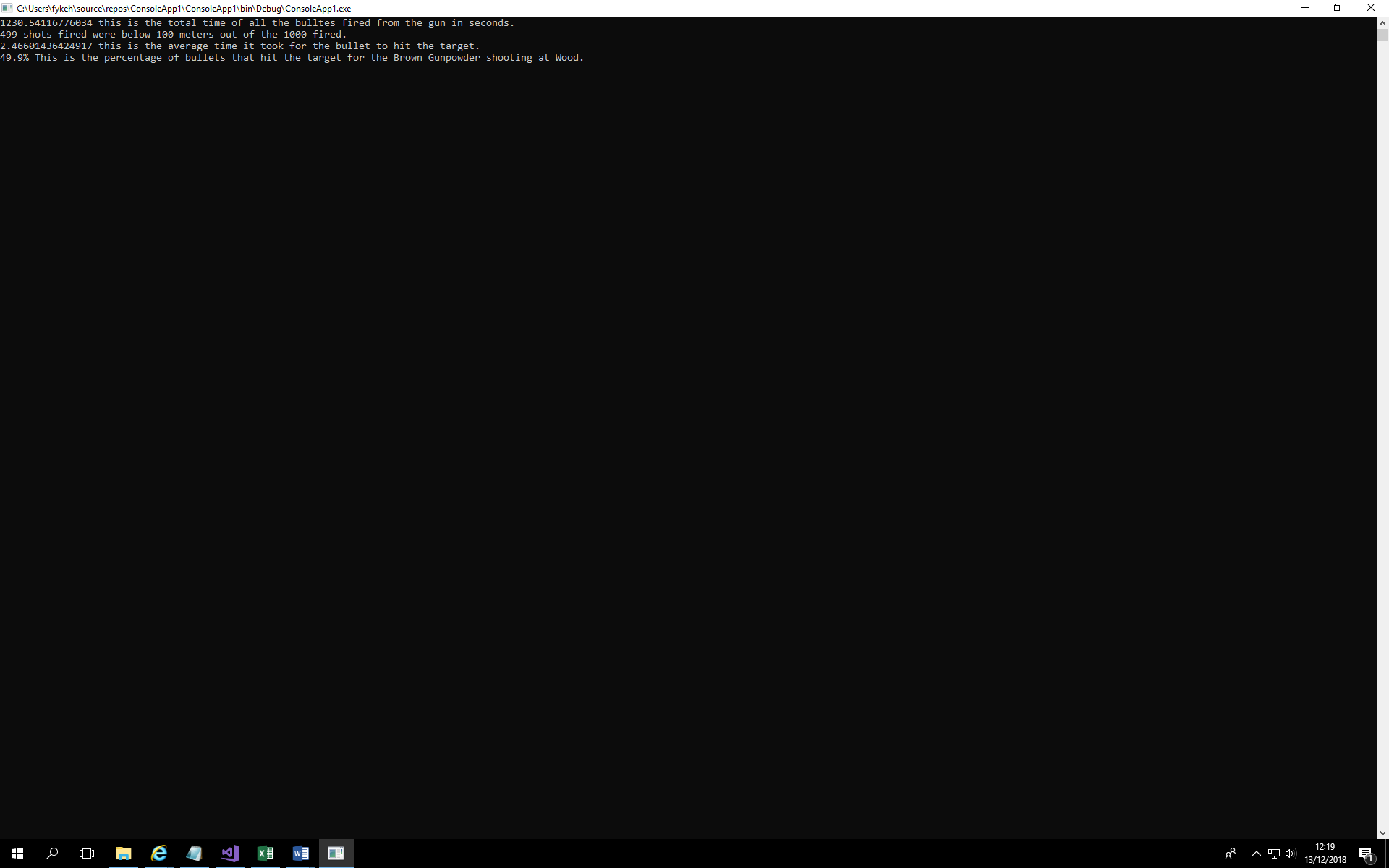
***Question 4:***

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*

***Word count***