**Task 2 Analysis and Design – Tom H**

**Introduction:**

The problem:

Compressing files down to a more functional scale is a concept that is incredibly useful when it comes to transferring large files around. When you compress a file with lossy compression you risk losing parts of the file when it returns to its original value so other compression methods will have to be used. Some files, such as video and audio, will use lossy compression because the data that is removed is usually indiscernible. With text files that contain sensitive data, you would want to use a form of compression otherwise the compression would cut out parts of it.

I need to develop a program that can identify all the unique words in a sentence and store these in a text file and uses this list to show only the positions of these words in the sentence.

How I intend to solve the problem:

I will make a program that can take a sentence and locate all the original words and replace them with a value that represents its position of unique occurrence in the text. It will store all the unique words and then show the sentence as a series of numbers that correlate to this list. Using only numbers to represent the words will cut down massively on the size of the file; if the file needs to be brought back to its original state it will be able to do so with ease and without any loss of data.

Success Criteria:

My success criterion for my program is:

* The program must be made in Visual Basic as it is a reliable programming language and it must include a user friendly design.
* Be able to save a file that contains text, which has been entered by the user from an input box.
* The program must be able to find all the unique words in the text and locate their original position by putting them in a list.
* The program must be able to replace the words in the sentence with the positions that will be linked to the unique words.
* The program must store all this information within a single text file.

Testing:

Whilst designing and coding my program I will take screenshots after significant bits of code and make notes of any problems I had in each part and how I overcame them. Once I think my program is sufficient I will design a test table that will take points from my success criteria, validation and more which I can then test against to see if the program works well enough without bugs or errors.

**Design:**

Inputs and Outputs:

*Inputs:*

* The sentence entered by the user.
* The file name entered by the user.

*Outputs:*

* A list of unique words in the text, all stored in a list.
* The text with the words replace with the initial positions of all the unique words in the text, all stored in a list.

User Interface:

\*See sheet of paper!\*

Visual Basic: Visual Basic is a simplistic coding language that allows me to use complex features to make programs fluent. In addition to this, Visual Studio -- the software in which I am coding in -- allows me to design a user interface that can look stylish and modern with a range of pleasant colours to make the program personal and my own.

Buttons: Buttons stand out in the program with text on top that gives the user a large sign, pointing towards what they need to press to do certain functions.

Labels: The title is displayed as a label and stands above the rest of the program telling the user exactly what the program is about.

List Box: I have used a list box in my program because I can display text easily in there without bringing it up as a message box which requires the user to click out of and forces them to remember given information or instructions. In a list box the user can look back on the history and double check anything that they make have forgotten instinctively.

Menu Strip: My program uses a menu strip as it can provide a range of features to a program all tucked away at the top. I can close the program or clear the text box easily or assign shortcuts to my program to increase the ease of access.

Flow Chart:

\*See sheet of paper!\*

Pseudocode:

*Form load:*

DECLARE *Sentence*

DECLARE *Word*

DECLARE *Sentence Array*

DECLARE *Unique Words*

DECLARE *Unique Positions*

DECLARE *Count*

*Sentence button:*

*Sentence* = **Input Box**

Has a *Sentence* been entered?

YES = OUTPUT *Sentence* to **lstTask1**

NO = OUTPUT **Error Message**

*Compress button:*

SPLIT *Sentence* => *Sentence Array*

LOOP THROUGH *Sentence Array*

*Word* = *Sentence Array (Count)*

Is the word in the unique words array?

YES = *Unique Words (Count) = Word*

YES = *Count* = *Count* + 1

END LOOP

LOOP THROUGH *Sentence Array*

*Word* = *Sentence Array (Count)*

LOOP THROUGH *Unique Words Array*

Is the word equal to the word at unique words (count)?

YES = *Unique Positions (Count)* = *Unique Word (Count)*

END LOOP

END LOOP

OUTPUT *Unique Words* TO A **Text File** AND a **List box**

OUTPUT *Unique Positions* TO A **Text File** AND a **List box**

*Reset button:*

CLEAR the **List Box**

*Exit button:*

CLOSE the **Program**

Validation:

My program needs to work efficiently without glitches so I need to have validation that will counteract any anomalies that may occur when the user is using the program which will include the following:

* Has a sentence been entered or not?
* Does the sentence contain a certain word more than once?
* Has a name been given to the file that was entered?

Variables:

*Sentence* = String

*Word* = String

*Unique\_Words* = Array

*Unique\_Positions* = Array

*Unique\_Words\_List* = String

*Unique\_Positions\_List* = String

*File\_Name* = String

*SW* = Stream Writer