Assignment 3: Cobol Programming / Re-engineering (20%)

Choose **one** of the following topics.

1. BABYLONIAN SQUARE ROOTS

The Babylonians didn't have any calculators, yet they were able to derive a recursive means of calculating square-roots. It is sometimes known as the *divide-and-average* algorithm. Here are the steps involved:

Step 1: Let N be the number where the square root is the be calculated, and R_0 represents an initial approximation, which could be any number. The closer to the actual value of the square root, the quicker the algorithm will work.

Step 2: Divide the number, N, by the approximation.

Step 3: Average the original approximation and the new approximation. For example:

$$R_1 = \frac{(R_0 + \frac{N}{R_0})}{2}$$

Step 4: Make the average value the new "approximation" and to go Step 2.

This process continues until the desired level of accuracy is achieved. Steps 2-4 can be calculated using the recursive relation:

$$R_k = \frac{(R_{k-1} + \frac{N}{R_{k-1}})}{2}$$

TASK

The task involves re-engineering an older version of a Cobol program to calculate square roots. There is no description of how the program works, just the code. The program works by reading in an ASCII file with a format specified in the program. Here is an example:

```
..........500000000100.....................................................
..........500000-00100.....................................................
......9184700000000100.....................................................
```

The periods represent spaces in the file. This will calculate the square roots of 5.0, -5.0, and 91847.0 .

The output is of the form:

```
          SQUARE ROOT APPROXIMATIONS
-------------------------------------------
         NUMBER                SQUARE ROOT
 ------------------      ------------------
             5.000000                2.236068
            -5.000000          INVALID INPUT
         91847.000000                303.062700
```

❶. Migrate the legacy Cobol program to a modern rendition of Cobol. The program contains a number of legacy features which should be removed, e.g. **go to** statements.

❷. Design a better way of obtaining input, i.e. remove the constraints of file input, and allow the user to calculate the square root of as many numbers as they wish. This means the interface should be interactive.

❸. Attempt to modularize the program more, by incorporating an external Cobol "function": `sqroot` to perform the actual calculation of the square root.

TOPIC 1 INFORMATION

REFLECTION DOCUMENT

Discuss your re-designed program in 3 page (or more if you like) *reflection document* (single-spaced), explaining decisions you made in the re-engineering process. Consider the document a synopsis of your experience with your Cobol re-engineering process - this should be at least 2 pages in length. This should include a synopsis of the approach you took to re-engineer the program (e.g. it could be a numbered list showing each step in the process). Identify the legacy structures/ features and how you modernized them. Keep track of the steps you used in the re-engineering process and document them, similar in fashion to the sample pracniques. Properly document the program, and explain its algorithmic history in your design document.

In addition, one page should describe your experience with Cobol, including answers to the following questions:

- Given your knowledge of programming, was Cobol easy to learn?
- What structures made Cobol challenging to program in?

DELIVERABLES

This submission should consist of the following items:

- The reflection document.
- The code (well documented and styled appropriately of course).

SKILLS

- This topic is an exercise in *migration* re-engineering, converting a program from one dialect of Cobol to another. It requires some research into the algorithm, and documentation of the re-engineered code.

2. THE TRITHEMIUS CIPHER

The Trithemius Cipher was developed by German abbot Johannes Trithemius in 1508 as an extension to the Caesar cipher. Trithemius is credited with publishing the first book on cryptology. His algorithm is based on the use of a *tabula recta* (from Latin), which is a square table of alphabets, each row of which is made by shifting the previous one to the left. It uses a *letter square* with the 26 letters of the alphabet following 26 rows of additional letters, each shifted once to the left, creating 26 different Caesar ciphers.

Therefore the plaintext "`let me`", will be encoded to the ciphertext "`lfvpi`" (spaces removed).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

The Trithemius letter square

It seemingly produces ciphertext that looks random, however is quite easy to break, as it lacks a key.

TASK

Write a Cobol program to implement the Trithemius Cipher.

❶. Your program should use two external Cobol "functions": `encrypt` and `decrypt` to perform encryption and decryption of a paragraph of text.

❷. The main program should prompt for an input file which contains the text to be converted. Output should be formatted to standard output.

TOPIC 2 INFORMATION

REFLECTION DOCUMENT

Discuss your program in 3 page (or more if you like) *reflection document* (single-spaced), explaining your algorithm and decisions you made in the process of designing your program. Consider the document a synopsis of your experience with Cobol - this should be at least 2 pages in length. The write-up should explain your approach to the problem, including any modifications made to the specifications. In addition you should include answers to the following questions:

- What were the greatest problems faced while designing the algorithm in Cobol?
- Was it easy to perform file I/O in Cobol?
- What particular features made Cobol a good/bad language?

In addition, one page should describe your experience with Cobol, including answers to the following questions:

- Given your knowledge of programming, was Cobol easy to learn?
- What structures made Cobol challenging to program in?

DELIVERABLES

This submission should consist of the following items:

- The design document.
- The code (well documented and styled appropriately of course).

SKILLS

- This topic involves writing a Cobol program from scratch, however it involves the use of external functions, and file I/O.