Assignment 3: Cobol Re-engineering (20%)

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
```

TASK 1

1. 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. The reengineered file should be called **`sqrtbaby.cob`**.

2. 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.

TASK 2

1. Attempt to modularize the program more, by incorporating an external Cobol "function": `squareroot` to perform the actual calculation of the square root. The main program for this task should be called **`sqrtbabyex.cob`**. The name of the external file is your choice (but it should be something appropriate).

WRITE-UP INFORMATION

REFLECTION REPORT

Discuss your re-designed program in 1 (one) page (or more if you like) **reflection report** (single-spaced), explaining decisions you made in the re-engineering process. Consider this document a synopsis of your experience with your Cobol re-engineering process. 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?
- What did you (i) like, (ii) dislike about Cobol?

DELIVERABLES

The submission should consist of the following items:

- The reflection report (PDF).
- The code (well documented and styled appropriately of course):
  `sqrtbaby.cob, sqrtbabyex.cob + external function file`

- Both the code and the reflection report should be submitted as a ZIP, TAR, or GZIP file.

SKILLS

- This 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.
- re-engineering, Cobol programming