Online Submission Assignment Cover Page

Name: Richard Break
ID#:
Date: 20/03/2012
Course #: CIS*3190
Course Name: Software for Legacy Systems
Instructor: Dr. Michael Wirth
Assignment #: 3
Assignment Name: COBOL Re-engineering
# of Pages (including this one): 4

	(For instructor's use) Grade:

Design Document

When updating the original Cobol Easter calculator, there were a number of changes required to allow for the original to compile with OpenCobol. Of these changes, there un-necessary lines of code that were removed, such as:

- AUTHOR. D E KNUTH.
- DATE-WRITTEN. JANUARY 22, 1962.
- DATE-COMPILED. JANUARY 23, 1962.
- SPECIAL-NAMES.
- FILE-CONTROL.

These examples were removed because they were either; obsolete and irrelevant towards the task the updated version would to accomplish, or modern coding conventions would have the fields as comments at the start of the file.

The next update to the original was the declaration of the variables contained within the file and working-storage sections. The original code contained lines such as:

- 03 YEARS; PICTURE IS ZZ999.
- 03 FILLER; SIZE IS 6 CHARACTERS.
- 77 TEMP; SIZE 6 NUMERIC COMPUTATIONAL.

That had to be updated to these lines, respectively:

- 03 years pic is zz999.
- 03 filler2 pic x(6).
- 77 temp pic 9(6).

These updates were necessary because OpenCobol does not recognize Cobol-61's formatting. Another update was the addition of the "easter-copy" record within the working-storage section. This was designed to hold the information on the dates of Easter before copied into the "easter-dates" record and written into the file to follow the Cobol-85 standard. Also the variable names "DAY" and "COLUMN" are predefined, and changed to "day-holder" and "column-number," respectively. Finally, the lines:

FD ANSWER-TABLE; LABEL RECORDS ARE STANDARD; DATA

RECORD IS EASTER-DATES.

01 EASTER-DATES.

Were updated to:

fd answer-table.

01 easter-dates.

As the statements “LABEL RECORDS” and “DATA RECORD” are obsolete in the Cobol-85 standard.

After these simple changes were completed, the task of updating the “procedure division” was the only task remaining. In Cobol-61, when a paragraph completed, it would then continue to run the code in the next paragraph, unless a go-to statement, the end of the file, or the end of the current section was met. In Cobol-85, this is not the case, as it would then return from the paragraph to the “perform” statement. If a paragraph wished to continue into another paragraph, the use of a go-to or perform statement would be necessary. Since go-to statements result in unstructured code, perform statements were used. Another issue contained within the procedure division was that the “control section.” and “computation section.” were not applicable to Cobol-85. This meant that all the updated paragraphs were to be written directly within the “procedure division” without any sections.

There were a few general updates required for the original code to meet Cobol-85’s standards, such as:

- Terminating “if” statements with “end-if”
- Terminating a perform loop with “end-preform”
- Updating old Relational Conditions such as “equals” and “exceeds” to “equal” and “greater”

The three paragraphs “outer-loop”, “middle-loop” and “inner-loop” were mostly maintained. To produce the same output as the original without the addition of new code, the values for each loop were kept the same. Within the inner-loop the main update was the write statements. They were updated so that they are directed to a file instead of a printer.

The paragraphs “fudge-epact”, “make-day-Sunday” and “transfer-answer” were merged with other existing paragraphs to reduce the number of control jumps within the program, which simplifies the program. The original code had multiple statements on a single line, such as:

DIVIDE 100 INTO YEAR GIVING CENTURY; ADD 1 TO CENTURY.

This was updated to two separate lines for legibility purposes

divide 100 into year giving century.

add 1 to century.

Another update was how to get a remainder from a division. The original code to do this was:

```
DIVIDE 30 INTO TEMP GIVING TEMP-1;

MULTIPLY 30 BY TEMP-1;

SUBTRACT TEMP-1 FROM TEMP;
```

This translates into one line of code, which is not only just shorter, but is more comprehensible:

```
divide 30 into temp giving temp-1 remainder temp.
```

The last topic to discuss is proper commenting of code. COBOL is design to be "self-documenting," which results in a reduction of the amount of commenting that is required to interpret the code easily. However, each paragraph contains a description of what operations are contained. Overall, the lack of documentation within the program is because it is intuitive.

How to compile:

Compile using OpenCobol.

cobc -x -free -Wall easter.cob

Then execute the program. The output is written in results.dat.