Assignment 3: Cobol Re-engineering (20%)

ROMAN NUMERAL CONVERTER

This assignment involves a program to convert a number expressed using Roman numerals to its decimal equivalent. The algorithm uses the Roman numerals I, V, X, L, C, D, and M - all others are considered invalid. The decimal equivalents of the above numerals are 1, 5, 10, 50, 100, 500, and 1000.

1. The string is scanned from left to right.
2. If the magnitude of the decimal equivalent of the *i*th numeral R_i is greater than or equal to the magnitude of the decimal equivalent of the $(i+1)$th numeral R_{i+1}, then the decimal equivalent R_i is added to the sum, otherwise R_i is subtracted from the sum.

For example, consider the task of converting the Roman numeral CXIV:

```
CXIV → 100
CXIV → 100 + 10
CXIV → 110 - 1
CXIV → 109 + 5
CXIV → 114
```

TASK

The task involves re-engineering an older version of a Cobol program to convert Roman numerals to their decimal equivalent. The code given to perform the Roman numeral conversion is written in COBOL, and consists of **two** files. This is Cobol's form of modularization - functions are always external to the main program. There is no description of how the program works, just three algorithmic flowcharts.

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. Also attempt to modularize the program more.

❷. One of the biggest problems with the algorithm is the way it takes input. Currently, to convert the number XVI, it has to be entered in the following manner:

```
X
V
I
<space>
```

This is somewhat tiresome. A better way of course would be to have the input formatted in a linear fashion:

```
XVI
```

Modify the program to allow for inputting Roman numerals in a linear fashion.

❸ The original program works on keyboard-based user input. Modify the program so it is possible to prompt for a filename containing a series of Roman numerals to be converted.

❹. Modify the program to allow either lowercase or uppercase Roman numerals.

DESIGN DOCUMENT

Discuss your re-designed program in 4-5 page design document, explaining decisions you made in the re-engineering process. 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.

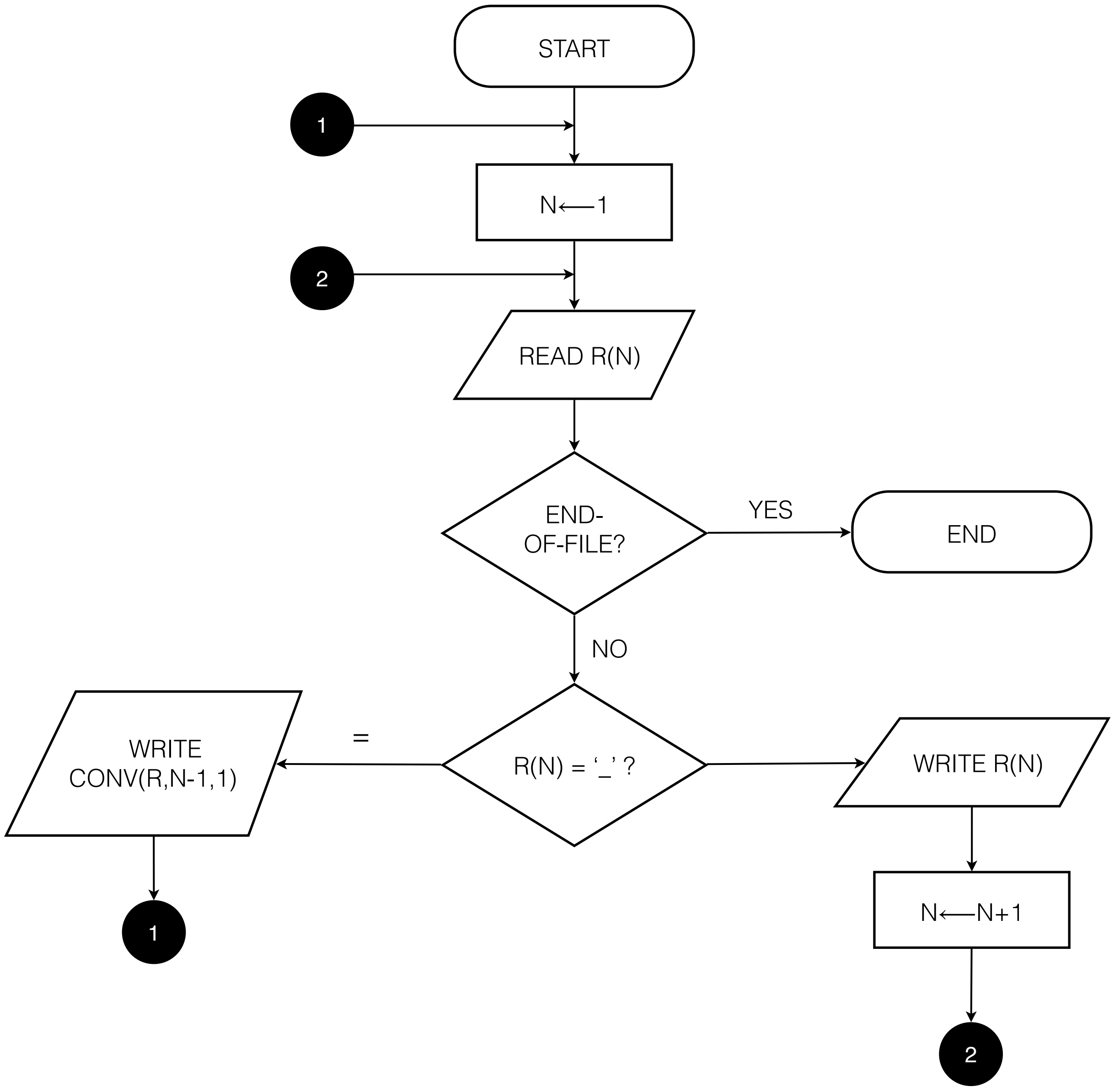
SKILLS

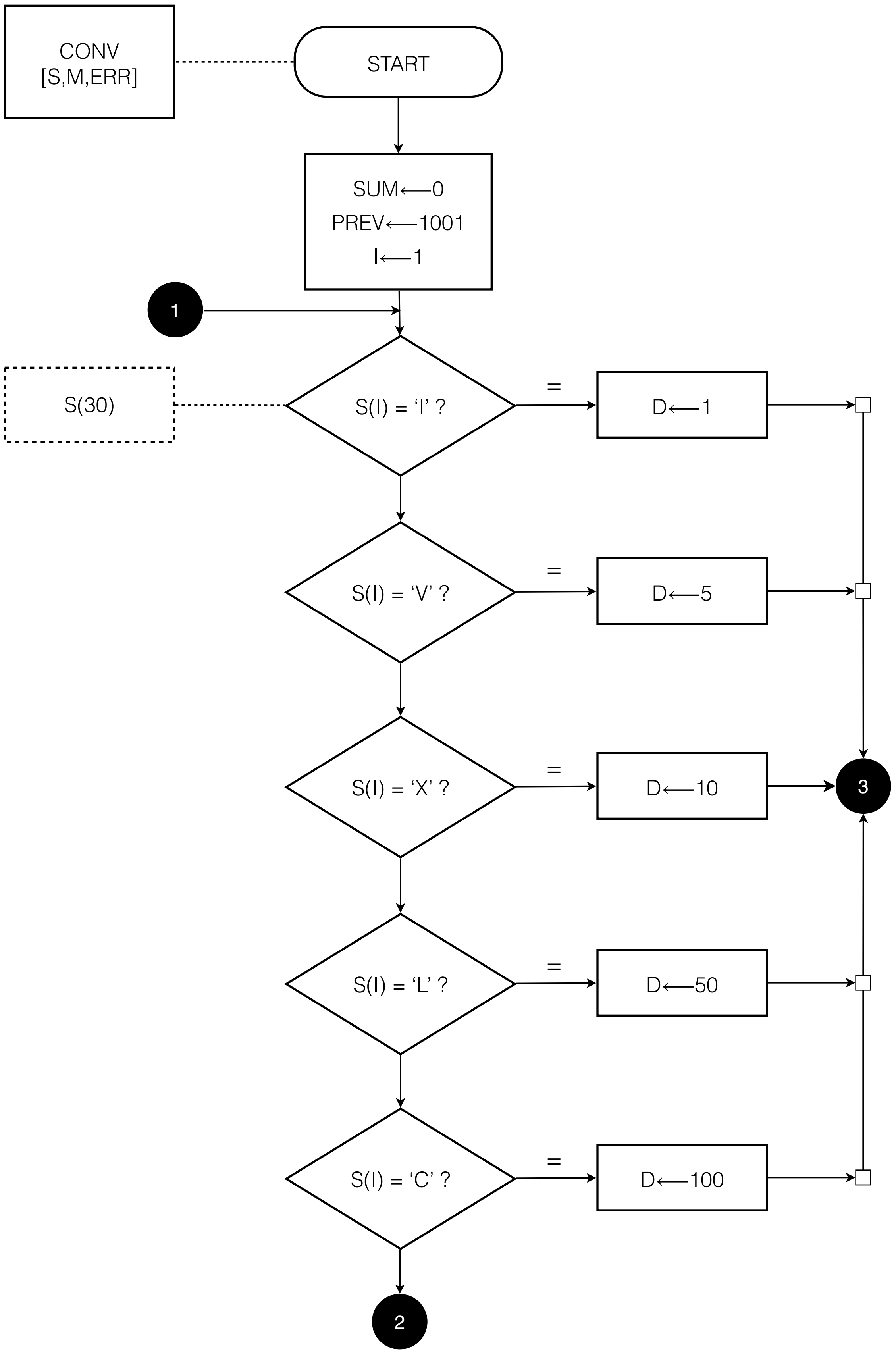
This is an exercise in *migration* re-engineering, converting a program from one dialect of Cobol to another. It requires some research into the algorithm, and documentation of the re-engineered code.

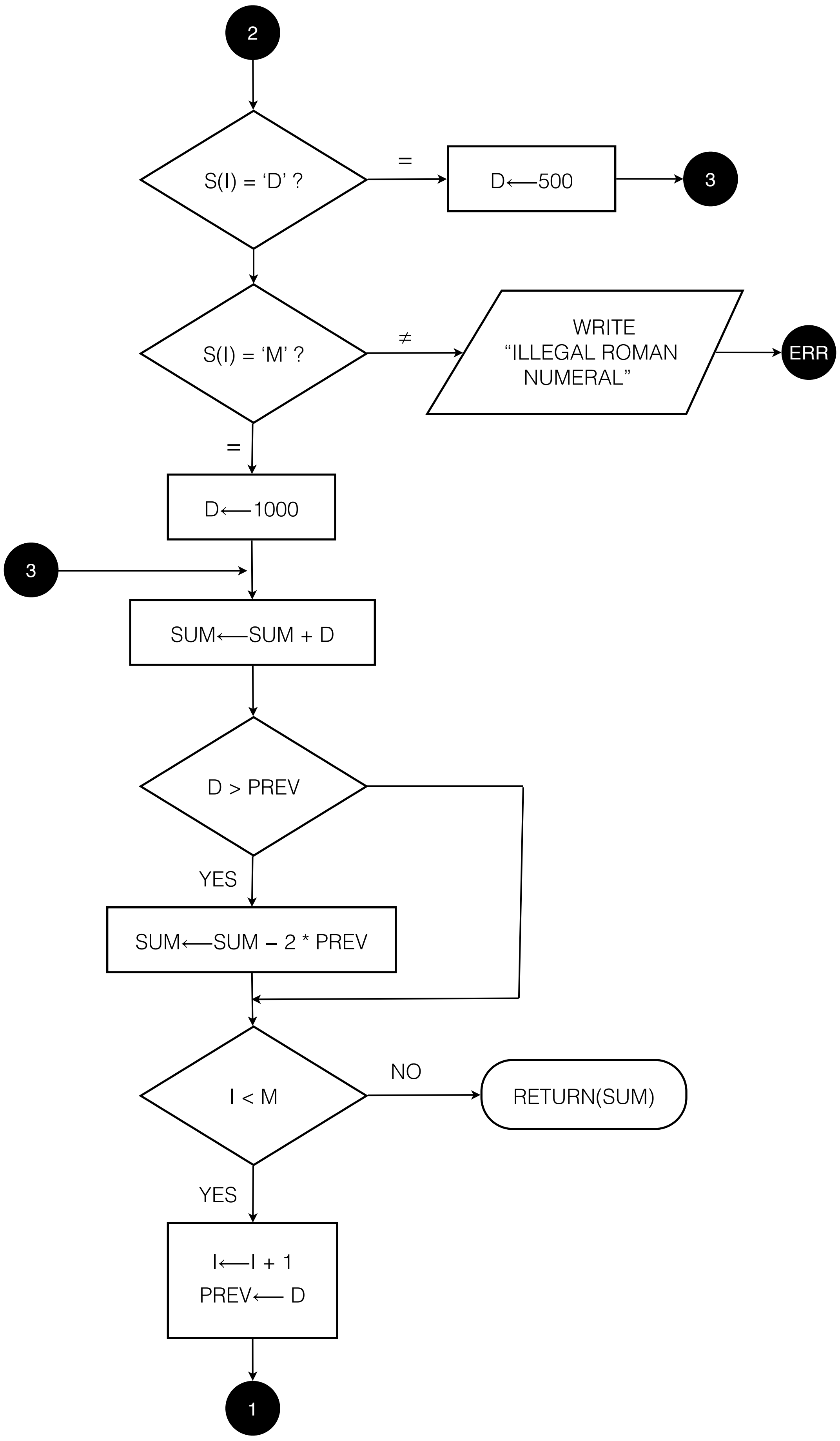
DELIVERABLES

Either submission should consist of the following items:

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


START
1
N⟵1
2
READ R(N)
END-
OF-FILE?
YES
END
NO
=
WRITE
CONV(R,N-1,1)
R(N) = '_' ?
WRITE R(N)
N⟵N+1
1
2


CONV
[S,M,ERR]
START
SUM⟵0
PREV⟵1001
I⟵1
1
S(30)
S(I) = ‘I’ ?
=
D⟵1
S(I) = ‘V’ ?
=
D⟵5
S(I) = ‘X’ ?
=
D⟵10
3
S(I) = ‘L’ ?
=
D⟵50
S(I) = ‘C’ ?
=
D⟵100
2


2
S(I) = ‘D’ ?
=
D⟵500
3
S(I) = ‘M’ ?
≠
WRITE “ILLEGAL ROMAN NUMERAL”
ERR
=
D⟵1000
3
SUM⟵SUM + D
D > PREV
YES
SUM⟵SUM − 2 * PREV
I < M
NO
RETURN(SUM)
YES
I⟵I + 1
PREV⟵ D
1