OzoBlockly Challenge: Use Evo to Convert Numbers to Roman Numerals

Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

Introduction

Roman numerals are commonly found on clock faces to represent hours, monuments and buildings for construction year, and copyright dates on movie and television title screens. Their origin dates back to ancient Rome, and modern usage makes use of seven symbols with integer values as follows: I (1), V (5), X (10), L (50), C (100), D (500), and M (1000). Thus, a clock might have a face with XII for Noon and Midnight. A building constructed in 1924 might have a cornerstone inscribed with MCMXXIV, and a movie released in 2018 might have a trailer screen with the Roman numerals MMXVIII.

The notation IV represents 4 as one less than five. The notation CM represents 900 as a hundred less than a thousand. Such notation is referred to as being subtractive. Subtractive notation is also used for 9 (IX) one less than 10, for 40 (XL) ten less than fifty, for 90 (XC) ten less than 100, and for 400 (CD) one-hundred less than five-hundred. Additive notation is used for numbers like 28 (XXVIII), where the three I's to the right of V are added to 5 to give 8. It should be mentioned that there are numerous variant forms for forming Roman numerals, but the most common form, discussed above, is what will be used throughout this challenge lesson.

The Challenge

In this challenge you will design an OzoBlockly program for which Evo converts a decimal number to Roman numerals. Since the largest number supported by Evo is 127, the largest Roman numeral needed here will be C (100). D and M will not be needed. There are a variety of algorithms that can be used to handle conversion to Roman numerals. However, you will be expected to make use of the algorithm that appears in the next section of this document.

Algorithm to Convert Numbers to Roman Numerals

Suppose that the decimal number that you wish to convert to Roman numerals is *x.*

1. Using the table of Figure 1, find the largest value *v* that is less than or equal to *x*. Jot down the corresponding largest Roman numeral *n*.

v	1	4	5	9	10	40	50	90	100	127
n	I	IV	V	IX	X	XL	L	XC	C	-----

Figure 1

2. Subtract the value *v* from *x*, giving a new value for *x*: $x = x - v$.
3. Repeat steps 1 and 2 until *x* equals 0.

Examples

Figure 2 shows four examples that help clarify how the algorithm works. Note how the entries in the *Result* column are obtained by concatenating the previous result with the next **n**. The entry in the bottom right cell of each table is the Roman numeral equivalent of the decimal number.

Example 1: x = 37				
Iteration	***x***	***v***	***n***	***Result***
1	37	10	X	X
2	27	10	X	XX
3	17	10	X	XXX
4	7	5	V	XXXV
5	2	1	I	XXXVI
6	1	1	I	**XXXVII**

Example 2: x = 94				
Iteration	***x***	***v***	***n***	***Result***
1	94	90	XC	XC
2	4	4	IV	**XCIV**

Example 3: x = 126				
Iteration	***x***	***v***	***n***	***Result***
1	126	100	C	C
2	26	10	X	CX
3	16	10	X	CXX
4	6	5	V	CXXV
5	1	1	I	**CXXVI**

Example 4: x = 74				
Iteration	***x***	***v***	***n***	***Result***
1	74	50	L	L
2	24	10	X	LX
3	14	10	X	LXX
4	4	4	IV	**LXXIV**

Figure 2

It should be noted that the algorithm does not work for Evo's largest number 127, as there is no corresponding Roman numeral in Figure 1 for 127. Therefore, the largest decimal value that will work with our algorithm is 126.

The Ozomap for the Roman Numerals Challenge

Figure 3 shows a reduced size Ozomap for discussion purposes here. A larger version suitable for printing for use by students can be found on the last page of this document. Evo is placed at the location labeled "Start" facing the direction shown by the arrow. Evo then moves to the first intersection where he speaks a random number from 1 through 126, inclusive. He then moves to the loop at the bottom of the Ozomap where he generates the Roman numeral that corresponds to the random number. For example, suppose that the random number is 49. On the first pass around the loop, Evo would stop and beep at the XL (40) intersection. On the second pass Evo would stop and beep at the IX (9) intersection. He would then turn off as he has completed the task of converting 49 to XLIX. Take a look at the accompanying video to see Evo convert 96 to its Roman numeral XCVI.

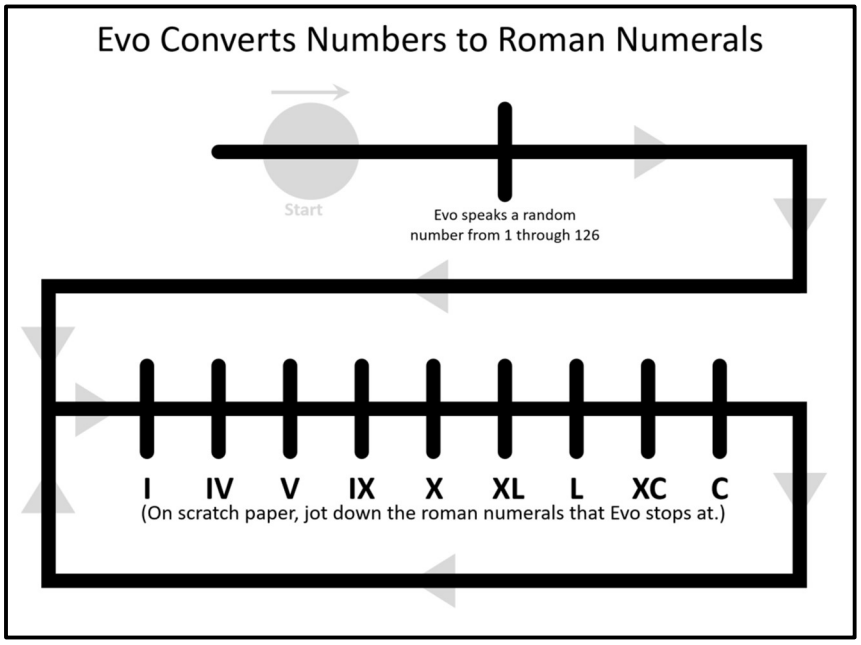

Figure 3

Challenge Requirements

1. Use the algorithm described on page 1 to convert from decimal numbers to Roman numerals.
2. Declare an array of length 10 to hold the *v* values of the table of Figure 1.
3. Define a function in which the 10 *v* values are stored in the array.
4. Define a function that searches the array for the largest value of v that is less than or equal to x. The function should have the current value of x as input and should return the index *i* into the array for the corresponding Roman numeral.
5. The index *i* can be used to help determine the intersection where Evo needs to stop and beep.

Evo Converts Numbers to Roman Numerals