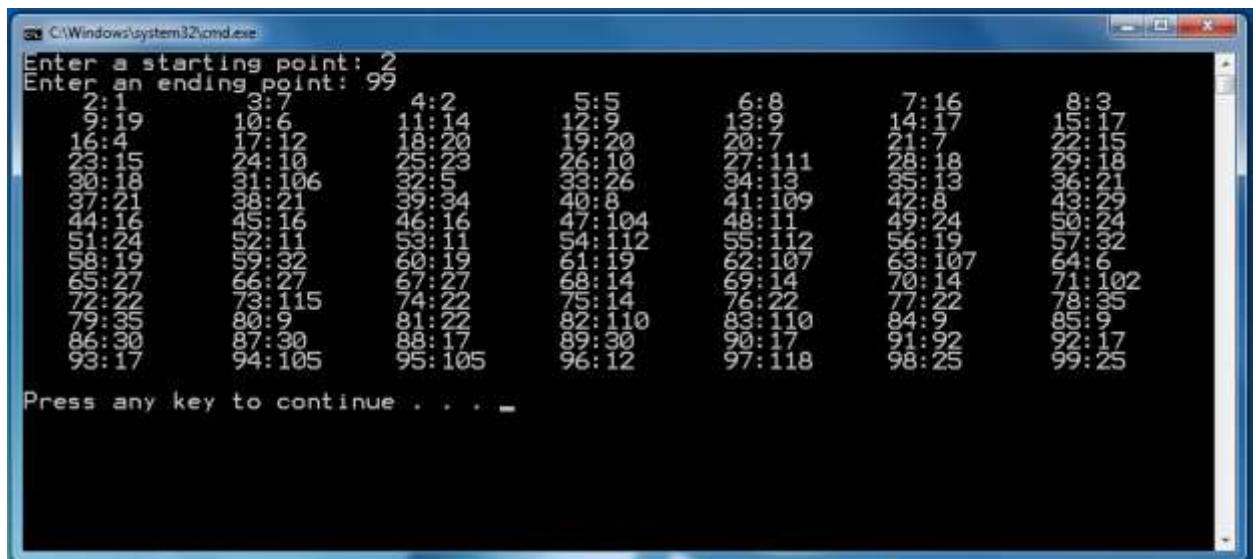


# The Collatz Conjecture

According to Wikipedia: The **Collatz conjecture** is a [conjecture](#) in [mathematics](#) named after [Lothar Collatz](#), who first proposed it in 1937. The conjecture is also known as the  **$3n + 1$  conjecture**, the **Ulam conjecture** (after [Stanisław Ulam](#)), **Kakutani's problem** (after [Shizuo Kakutani](#)), the **Thwaites conjecture** (after Sir Bryan Thwaites), **Hasse's algorithm** (after [Helmut Hasse](#)), or the **Syracuse problem**.<sup>[1][2]</sup> the sequence of numbers involved is referred to as the **hailstone sequence** or **hailstone numbers** (because the values are usually subject to multiple descents and ascents like [hailstones](#) in a cloud),<sup>[3][4]</sup> or as **wondrous numbers**.<sup>[5]</sup>

Take any [natural number](#)  $n$ . If  $n$  is even, divide it by 2 to get  $n/2$ . If  $n$  is odd, multiply it by 3 and add 1 to obtain  $3n + 1$ . Repeat the process (which has been called "Half Or Triple Plus One", or **HOTPO**<sup>[6]</sup>) indefinitely. The conjecture is that no matter what number you start with, you will always eventually reach 1. The property has also been called **oneness**.

Create a project called Program1. Add a C source file to the project named program1.c. Your program should create a table of numbers providing information about the number of steps that a particular number takes in order to reach the number 1 in the sequence described above. Your program should create the table by first asking the user to enter a starting integer  $> 1$  and less than 1000 (ONE THOUSAND) and then an ending number that is greater than their starting number and less than 10000 (TEN THOUSAND). For every integer in between, and including the start and end points, your program should compute the number of steps it takes to reach the number 1 and display it beside the original number. A sample run with the starting point of 2 and the ending point of 99 is given below:



```

C:\Windows\system32\cmd.exe
Enter a starting point: 2
Enter an ending point: 99
 2:1    3:7    4:2    5:5    6:8    7:16    8:3
 9:19   10:6   11:14   12:9   13:9   14:17   15:17
16:4   17:12   18:20   19:20   20:7   21:7   22:15
23:15   24:10   25:23   26:10   27:111  28:18   29:18
30:18   31:106   32:5    33:26   34:13   35:13   36:21
37:21   38:21   39:34   40:8    41:109   42:8    43:29
44:16   45:16   46:16   47:104   48:11   49:24   50:24
51:24   52:11   53:11   54:112   55:112   56:19   57:32
58:19   59:32   60:19   61:19   62:107   63:107   64:6
65:27   66:27   67:27   68:14   69:14   70:14   71:102
72:22   73:115   74:22   75:14   76:22   77:22   78:35
79:35   80:9    81:22   82:110   83:110   84:9    85:9
86:30   87:30   88:17   89:30   90:17   91:92   92:17
93:17   94:105   95:105   96:12   97:118   98:25   99:25

Press any key to continue . . . _
  
```

Your program should print a maximum of 7 columns per line where each column follows the following format rules: The first number is printed right justified in 5 spaces then a colon is printed then the number of steps is written as a left justified number in 5 spaces. The end result is that we get columns centered on the colon character as shown in the screenshot.

You should make use of functions to make your code easy to read.

At the top of your program you should have a comment section that follows the below format:

```
/******  
    Author: <insert your name>  
    Date: <insert today's date>  
  
    Purpose: <Insert a short description of what  
              your program does here.>  
    Time Spent: <Insert how much time you spent  
                on the assignment here>  
*****/
```