## Problem 4 – Magic Dates

Consider we are given a **date** in format dd-mm-yyyy, e.g. 17-03-2007. We calculate the **weight of this date** by joining together all its digits, multiplying each digit by each of the other digits and finally sum all obtained products. In our case we will have 8-digits: 17032007, so the weight is 1\*7 + 1\*0 + 1\*3 + 1\*2 + 1\*0 + 1\*0 + 1\*7 + 7\*0 + 7\*3 + 7\*2 + 7\*0 + 7\*0 + 7\*7 + 0\*3 + 0\*2 + 0\*0 + 0\*0 + 0\*7 + 3\*2 + 3\*0 + 3\*0 + 3\*7 + 2\*0 + 2\*0 + 2\*7 + 0\*0 + 0\*7 + 0\*7 = 144.

Your task is to write a program that finds all **magic dates**: **dates between two fixed years matching given magic weight**. The dates should be printed in increasing order in format dd-mm-yyyy. We use the traditional calendar (years have 12 months, each having 28, 29, 30 or 31 days, etc.)

### Input

The input data should be read from the console. It consists of 3 lines:

* The first line holds an integer number – **start year**.
* The first line holds an integer number – **end year**.
* The first line holds an integer number – **magic weight**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output should be printed on the console as a sequence of dates in **format dd-mm-yyyy** in **alphabetical order**. Each string should stay on a separate line. In case no magic dates exist, print “**No**”.

### Constraints

* The **start** and **end year** are **integers** in the range [1900-2100].
* The **magic weight** is an integer number in range [1…1000].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 2007  2007  144 | 17-03-2007  13-07-2007  31-07-2007 | 2003  2004  1500 | No | 2012  2014  80 | 09-01-2013  17-01-2013  23-03-2013  11-07-2013  01-09-2013  10-09-2013  09-10-2013  17-10-2013  07-11-2013  24-11-2013  14-12-2013  23-11-2014  13-12-2014  31-12-2014 | 2011  2012  14 | 01-01-2011  10-01-2011  01-10-2011  10-10-2011 |