Lab session 2: Imperative Programming

This is the second lab of a series of weekly programming labs. You are required to solve these programming exercises and submit your solutions to the automatic assessment system *Themis*. The system is 24/7 online. Note that the weekly deadlines are very strict (they are given for the CET time zone)! If you are a few seconds late, Themis will not accept your submission (which means you failed the exercise). Therefore, you are strongly advised to start working on the weekly labs as soon as they appear. Do not postpone making the labs! You can find Themis via the url `https://themis.housing.rug.nl`

Note: In lab session 2 you are only allowed to make use of constructs that are taught in the weeks 1 and 2 (so no arrays, functions, recursion, etc).

Problem 1: Divisible by 3

You probably see immediately that 12 is divisible by 3. But what about a number like 12345678? It is much harder to see that it is ($12345678 = 3 \times 4115226$). There is a nice trick to quickly find out whether a (large) number is divisble by 3 (without using a calculator).

A positive integer is divisible by 3 if and only if the sum of all its digits is divisible by 3.

Of course, we can repeat this process until we reach a number less than 10 (i.e. a single digit number). As an example we have another look at the number 12345678. The sum of its digits is $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36$. We repeat the process one more step by adding up the digits of 36 yielding $3 + 6 = 9$. Since 9 is a single digit number the process stops. Clearly, 9 is divisible by 3 so we now know that 12345678 is divisible by 3.

Write a program that reads from the input an integer n (where $n > 9$) and outputs the steps that this method makes to determine whether n is divisible by 3 or not, and the final conclusion (`YES` or `NO`). Make sure that your output matches the outputs given in the following examples (beware of spaces and newlines!). Your program is not allowed to use the remainder/modulo operation to determine whether a number is divisible by 3 (i.e. the `%` operator). Your program is only allowed to use the modulo operator to produce the sum of digits in each subsequent step.

Example 1:

input:
```
9999
```
output:
```
9999 -> 36 -> 9
YES
```

Example 2:

input:
```
987
```
output:
```
987 -> 24 -> 6
YES
```

Example 3:

input:
```
445
```
output:
```
445 -> 13 -> 4
NO
```

Problem 2: Divisible by 7

There is also an interesting trick to determine whether a positive integer n is divisible by 7. If $n < 7$ then n is not divisible by 7. If $n = 7$ or $n = 49$ then it clearly is divisible by 7. For any other n, split n in two numbers p and q, where p is the number which is obtained by removing the last digit of n and q is the last digit itself. Next, we compute $n' = p + 5 \cdot q$. If n' is divisible by 7, then so is n. Of course, we can repeat this process until $n' = 49$ or $n' \leq 7$. You do not have to understand why this algorithm works (especially termination is tricky), and may simply assume that it works (trust us, it does!).
For example, if we want to investigate whether 17 is divisible by 7 then we need the following 16 steps to conclude that it is not:

$$17 \rightarrow 36 \rightarrow 33 \rightarrow 18 \rightarrow 41 \rightarrow 9 \rightarrow 45 \rightarrow 29 \rightarrow 47 \rightarrow 39 \rightarrow 48 \rightarrow 44 \rightarrow 24 \rightarrow 22 \rightarrow 12 \rightarrow 11 \rightarrow 6.$$

Write a program that reads from the input an integer n (where $n > 0$) and outputs the number of steps that this method makes to determine whether n is divisible by 7 or not, and the final conclusion (`YES` or `NO`). Make sure that your output matches the outputs given in the following examples. Again, your program is not allowed to use the remainder/modulo operation to determine whether a number is divisible by 7 (i.e. the `%` operator). Your program is only allowed to use the modulo operator to form the subsequent n' numbers.

Example 1:
input:
```
17
```
output:
```
16
NO
```

Example 2:
input:
```
777
```
output:
```
3
YES
```

Example 3:
input:
```
776
```
output:
```
12
NO
```

Problem 3: Solving Equations

The input of this problem consists of 5 positive integers m, n, p, q, and r. The output should be the number of possible combinations of non-negative integer values a, b, c such that:

$$a + b + c = m \text{ and } a \cdot p + b \cdot q + c \cdot r = n.$$

For example, for $m = 10$, $n = 1000$, $p = 80$, $q = 60$, and $r = 120$ there are only two solutions, being $(a, b, c) = (2, 2, 6)$ and $(a, b, c) = (5, 0, 5)$.

Write a program that reads from the input the values m, n, p, q, r and outputs the number of combinations (a, b, c) (where a, b, c are non-negative integers) that satisfy the above equations. Note that (very) inefficient programs will fail Themis' judgment.

Example 1:
input:
```
100 10000 99 88 102
```
output:
```
5
```

Example 2:
input:
```
10 1000 80 60 120
```
output:
```
2
```

Example 3:
input:
```
50 2500 25 50 100
```
output:
```
17
```

Problem 4: Sum of Prime Numbers

The input for this problem is a series of positive integers terminated by the value zero. The output of your program should be the sum of all prime numbers in this series. You may assume that the sum fits in a standard integer (`int`). Note that (very) inefficient programs will fail Themis' judgment.

For example, for the input series `5 3 8 0` the output should be 8 (because 5 and 3 are prime numbers).

Example 1:

input:
```
5 3 8 0
```
output:
```
8
```

Example 2:

input:
```
5 3 2 12 8 7 17 0
```
output:
```
34
```

Example 3:

input:
```
147 29 3 19 37 0
```
output:
```
88
```