Lab session 3: Imperative Programming

This is the third 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 3 you are only allowed to make use of the conditional constructs, loops and functions (so no arrays, etc). You are not allowed to use the math library.

Problem 1: Function Evaluator

Given are the following three functions on integers:

$$f(x) = 3x + 1 \quad g(x) = x/2 \quad h(x) = x^2 - x$$

Implement these functions in C. Next, write a program that accepts on its input an integer n followed by a series of letters f, g, and h terminated by an equals sign (=). The output should be the number that is the output that is obtained by applying the given sequence of functions to the input number. For example, the calculation of $f(g(h(42)))$ is specified by the input `42 hgf=`. The output should be 2584 because $f(g(h(42))) = f(g(1722)) = f(861) = 2584$.

Example 1:
input:
```
42 hgf=
```
output:
```
2584
```

Example 2:
input:
```
12 fghg=
```
output:
```
153
```

Example 3:
input:
```
1 fff=
```
output:
```
40
```

Problem 2: Special Odd Numbers

The input consists of two integers n and m. The task is to find the total amount of numbers from n up to and including m which:

1. are prime,
2. consist of only odd digits,
3. have an odd number of ones in its binary representation.

For example 13 is a number that satisfies the properties since, it is a prime, consists of only odd digits and the binary representation holds an odd number of ones `[1,1,0,1]`. Another example is 17 which is not a special odd number since the binary representation holds an even number of ones `[1,0,0,0,1]`. For each of the conditions above write a separate function.

Example 1:	**Example 2:**	**Example 3:**
input:	**input**:	**input**:
`2 10`	`10 50`	`100 1000`
output:	**output**:	**output**:
`1`	`5`	`24`

Problem 3: Palindromic Divisors

A number is called to be palindromic if they read the same backwards and forwards. An example of such a number is 131. In this problem, the task is to find numbers whose only palindromic divisor is 1.

One example of such a number is 13. It is simple to see that the only divisors of 13 are 1 and 13. Only 1 is a palindromic number, so therefore 13 is a number that satisfies the description. Another example is 169. 169 can be divided by 1, 13 and 169. Again only 1 is a palindromic number, hence 169 satisfies the description. A number that does not fit the description is 2. Since 2 can be divided by 1 and 2 and are both palindromic numbers.

Write a program that reads from the input two integers a and b. The output should be the number of integers in the range a up to and including b whose only palindromic divisor is 1. Before starting the exercise, think about how you would separate the functionality in different functions.

Example 1:	**Example 2:**	**Example 3:**
input:	**input**:	**input**:
`1 10`	`20 50`	`1 100`
output:	**output**:	**output**:
`1`	`7`	`21`

Problem 4: Primonacci Numbers

In the lecture we discussed the famous Fibonacci series which is defined as follows:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}.$$

The Primonacci series is a variation on this pattern. It is defined as follows:

$$D_0 = 2, \quad D_1 = 3, \quad D_n = f(D_{n-1} + D_{n-2}),$$

where $f(x)$ yields the prime number $p >= x$ closest to x. The input for this problem is an integer n, where $0 \leq n \leq 42$. The output must be the number D_n. Note that it is not allowed to use precomputed answers (apart from the starting values 2 and 3) in your program code. The teaching assistants will check this manually. Again, think about how you would separate the functionality in different functions.

Example 1:
input:
```
3
```
output:
```
11
```

Example 2:
input:
```
1
```
output:
```
3
```

Example 3:
input:
```
5
```
output:
```
29
```