# 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 the conditional constructs and loops (so no arrays, functions, etc). You are not allowed to use the math library.**

# Problem 0: Assignments

For each of the following annotations determine which choice fits on the empty line (.....). The variables `x`, and `y` are of type `int`. Note that `A` and `B` (uppercase letters!) are specification constants (so not program variables). After determining which choice fits, work out the intermediate steps and submit a single pdf with your solutions through Themis.

1.1
```
/* x == A */
.....
/* x == 7*A + 5 */
```

(a) `x = 7*x + 5;`
(b) `x = x/7 - 5;`
(c) `x = (x - 5)/7;`

1.2
```
/* x == A, y == B */
.....
/* x == B - A, y == A */
```

(a) `x = x - y; y = y - x;`
(b) `x = y - x; y = y - x;`
(c) `x = y - x; y = x - y;`

1.3
```
/* x - 1 == B, y == x * x + A */
.....
/* x == B, y == x * x + A */
```

(a) `x = x - 1; y = y - 2*x - 1;`
(b) `x = y + 1; y = y + 2*x + 1;`
(c) `x = x - 1; y = y - 2*x + 1;`

1.4
```
/* x == A, y == B */
x = 3*y + x; y = 3*y - x; x = x + y;
.....
```

(a) `/* x == 3*B, y == A */`
(b) `/* x == -3*B, y == A */`
(c) `/* x == 3*B, y == -A */`

1.5
```
/* x - y == B, y == A */
y = x - y; x = x - y;
.....
```

(a) `/* x == B, y == A */`
(b) `/* x == A, y == B */`
(c) `/* x == A, y == A */`

1.6
```
/* 2*x + 4*y - 2z > 4 */
x = x + 2*y + 1; z = x - z;
.....
```

(a) `/* z > 3 */`
(b) `/* z > 12 */`
(c) `/* 3*x + 6*y + 1 - z > 4 */`

# Problem 1: Reversible Divisors

In this exercise you are asked to find numbers for which both the number $n$ itself and its reverse are divisible by some number $d$. For example take the pair $(n, m) = (412, 2)$. Both 412 and 214 are divisible by 2, so the condition in clearly met. To make it a bit more interesting, you are also asked to find all numbers within a specified range from $a$ up to and including $b$ for which this property holds with divisor $d$. As an example, consider the range from 0 up to and including 20 with divisor 2. The condition is met for the numbers $2, 4, 6, 8$ and 20.

Write a program that reads from the input three integers $a$, $b$ and $d$. The integers $a$ and $b$ define the range and $d$ is the divisor. The output should be the number of integers in the range $a$ up to and including $b$ for which the numbers and its reverse are divisible by $d$. Note that (very) inefficient programs will fail Themis' judgment.

**Example 1:**

**input**:
```
0 20 2
```
**output**:
```
6
```

**Example 2:**

**input**:
```
20 40 3
```
**output**:
```
7
```

**Example 3:**

**input**:
```
100 300 4
```
**output**:
```
10
```

# Problem 2: Compelling Numbers

A compelling number is a number which has more divisors than the number of digits in its binary representation. An example of such a number is 6. The binary representation of 6 is `[1, 1, 0]`, which has *three* binary digits. It is easy to see that 6 has *four* divisors: `[1, 2, 3, 6]`.

Write a program that reads from the input two integers $a$ and $b$. The integers $a$ and $b$ define the range over which you need to find compelling numbers. The output should be the number of compelling numbers in the range $a$ up to and including $b$.

**Example 1:**

**input**:
```
0 20
```
**output**:
```
4
```

**Example 2:**

**input**:
```
150 200
```
**output**:
```
11
```

**Example 3:**

**input**:
```
0 100
```
**output**:
```
24
```

# Problem 3: Sum Unitary Divisors

The input consists of one integer $n > 1$. The output of the program indicates how many unitary divisors the number $n$ has. A number $m$ is a unitary divisor of $n$ if:

1. $m$ is a divisor of $n$,
2. the only common divisor for $m$ and $n/m$ is 1.

Take for example $n = 20$. All divisors of 20 are 1, 2, 4, 5, 10, 20. Consider each of these divisors as $m$ and compute $n/m$. This yields the following 6 pairs:

`(1, 20), (2, 10), (4, 5), (5, 4), (10, 2), (20, 1).`

The only common divisor for the pairs $(1, 20), (4, 5), (5, 4)$ and $(20, 1)$ is 1. The other two pairs $(2, 10)$ and $(10, 2)$ contain numbers which can both be divided by a value other than 1, hence not contributing to the number of unitary divisors. Since the properties hold for 4 pairs, the output of the program is 4.

**Example 1:**
**input**:
```
20
```
**output**:
```
4
```

**Example 2:**
**input**:
```
5
```
**output**:
```
2
```

**Example 3:**
**input**:
```
30
```
**output**:
```
8
```

# Problem 4: Electronic Devices

A company needs a set of electronic devices (`phones`, `laptops` and `tablets`) for their employees to do their work. The company has a certain budget $n$ available to purchase the devices. All employees receive the same phone, laptop or tablet so the prices are fixed. In order to make proper use of their budget, it is important to spend all the money available (i.e. there should be no money left). Consider a budget of 2220 euros to buy phones, laptops and tablets that cost respectively 210, 220 and 230 euros. There are 5 combinations that equal exactly 2220 euros.

1. (0 phones, 8 laptops, 2 tablets)
2. (1 phones, 6 laptops, 3 tablets)
3. (2 phones, 4 laptops, 4 tablets)
4. (3 phones, 2 laptops, 5 tablets)
5. (4 phones, 0 laptops, 6 tablets)

Write a program that reads from the input four integers $a$, $b$, $c$ and $n$. The first three integers are the prices of the phones, laptops and tablets and the last integer represents the available budget. On the output you should print the number of combinations that allows you to fully spend the budget. Note that (very) inefficient programs will fail Themis' judgment.

**Example 1:**
**input**:
```
210 220 230 2220
```
**output**:
```
5
```

**Example 2:**
**input**:
```
10 10 10 100
```
**output**:
```
66
```

**Example 3:**
**input**:
```
230 200 200 2000
```
**output**:
```
11
```