Lab session 1: Imperative Programming

This is the first 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`

All lab exercises assume that you are running Linux. If you do not, do not expect help from the teaching assistants in solving computer related problems. If you do not have Linux on your computer, you are strongly advised to install it. You can install it next to your current operating system (a so called dual boot system system) or you can run it in a virtual machine. If you do not have experience installing an operating system (or a dual boot system), then the safest option is to use a virtual machine. You can run a virtual machine (also on Apple Mac's) using the program *virtualbox*, which is freely available via `https://www.virtualbox.org/`. Once you installed virtualbox, you can install any Linux distribution. If you want to install Linux in a virtual machine yourself, then we advise the Ubuntu distribution that uses the XFCE desktop which is called *xubuntu* (see `https://xubuntu.org/`). However, we made a pre-configured virtual machine image which you can download from Nestor (this is a very large file, so downloading it will take some time).

Note: In lab session 1 you are only allowed to make use of constructs that are taught in week 1 (so no loops, arrays, functions, etc).

Problem 1: Hello world!

The first exercise is to make a small computer program that outputs `Hello world!` on the screen. The goal of the exercise is to get acquainted with the programming environment on a Linux system and the automatic assessment system *Themis*.

1. Open a terminal window. In this terminal you can type commands. Start by creating a directory `impprog` that you will use for the course Imperative Programming. You create this directory with the command:
   ```
   mkdir impprog
   ```
2. Navigate to this directory using the command `cd` (change directory).
   ```
   cd impprog
   ```
3. Create another directory `week1`, which is a subdirectory of `impprog`. Create in the directory `week1` another subdirectory `hello` and navigate to this directory:
   ```
   mkdir week1
   ```

```
cd week1
mkdir hello
cd hello
```

Create the file `hello.c` using an editor (for example `geany`).

```
geany hello.c &
```

Type the following program, replace `...` by the right information, and save it.

```
/* file:    hello.c              */
/* author:  ...... (email: ....) */
/* date:    ......               */
/* version: .....                */
/* Description: This program prints 'Hello world' */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char *argv[]) {
  printf("Hello world!\n");
  return 0;
}
```

4. Compile the program using the command:

   ```
   gcc -std=c99 -Wall -pedantic hello.c
   ```

5. On successful compilation, an executable file `a.out` is produced. Run the program:

   ```
   ./a.out
   ```

6. If you are convinced that your program works well, you can submit it to the online assessment system Themis. You have completed this first task once Themis accepted your submission.

Problem 2: Exam Hall

Due to the Covid-19 pandemic it is difficult to organize on-site exams in the exam hall. The board of the university has a hard time deciding whether exams can take place in the exam hall, or not. In their meeting, there is heavy debate about the floor space per student. They decide that each student must have a square space of $n \times n$, while the total floor size is $w \times h$ (where n, w, and h are positive integers). They are in need of a tool that (given the values of n, w, and h) computes how many students can do exam at the same time in the exam hall.

Write a program that expects on the input the three integers (first n, next w, followed by h) and outputs the number of students that can do an exam at the same time for this scenario. Note that your output must end with a newline (`\n`).

Example 1:

input:
```
5 50 50
```
output:
```
100
```

Example 2:

input:
```
10 50 50
```
output:
```
25
```

Example 3:

input:
```
3 30 90
```
output:
```
300
```

Problem 3: Ascending Numbers

A positive integer is called an *ascending number* if, read from left to right, its (decimal) digits are in ascending order. For example, the number 2334 is an ascending number because $2 \leq 3$, $3 \leq 3$, and $3 \leq 4$. Clearly, the number 423 is not an ascending number, because $4 > 2$.

Write a program that reads from the input an integer n (where $0 \leq n < 10000$) and outputs `YES` if n is an ascending number, or `NO` otherwise.

Example 1:

input:
```
2334
```
output:
```
YES
```

Example 2:

input:
```
423
```
output:
```
NO
```

Example 3:

input:
```
1234
```
output:
```
YES
```

Problem 4: Cigarettes

A poor homeless man can make exactly one cigarette out of four cigarette butts. One day he finds 31 butts. How many cigarettes can he smoke that day?

Well, from 31 butts he can make 7 cigarettes (using $7 \times 4 = 28$ butts, leaving 3 butts). So, after having smoked these 7 cigarettes he has 7 new butts, totalling $7 + 3 = 10$ butts, which is enough to make another 2 cigarettes (leaving $10 - 2 \times 4 = 2$ butts). After having smoked these two cigarettes, he again has two new butts. Together with the remaining 2 butts, he can make anoter (last) cigarette. So, in total he can smoke $7 + 2 + 1 = 10$ cigarettes.

Write a program that reads from the input a positive integer n, which is the number of butts that the man finds on some day. The output should be the number of cigarettes that he can smoke that day. You may assume that $0 \leq n < 1000000$. Note that you can solve this problem by doing some mathematical reasoning. You are not allowed to use a loop (or any other iterative construct) to solve this problem.

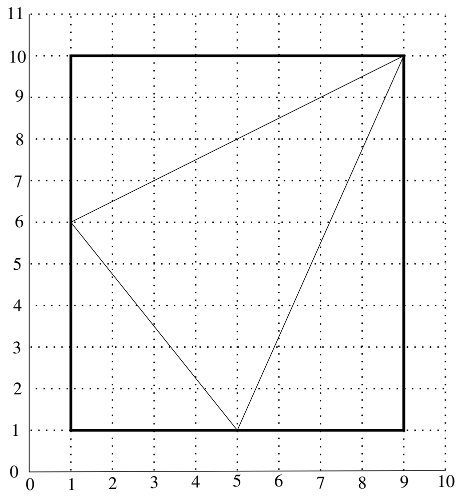
Example 1:
input:
```
3
```
output:
```
0
```

Example 2:
input:
```
4
```
output:
```
1
```

Example 3:
input:
```
31
```
output:
```
10
```

Problem 5: Bounding Box

In the figure on the right, you see a grid containing a triangle and a rectangle. The rectangle is the smallest axes-aligned rectangle that surrounds the triangle completely. This rectangle is called the *bounding box* of the triangle. The bottom left corner point of this bounding box is the grid point $(1, 1)$, and the top right corner is the point $(9, 10)$.



Make a program that reads from the input the three coordinates of the vertices of a triangle. You may assume that all input values are integers. The program must print on the screen the bottom left corner point and the top right corner point of the bounding box of this triangle. Make sure that your program produces output in the exact same format as given in the following examples. The first example corresponds with the figure.

Example 1 (see figure):
input:
```
1 6
5 1
9 10
```
output:
```
[(1,1),(9,10)]
```

Example 2:
input:
```
0 5
4 0
8 9
```
output:
```
[(0,0),(8,9)]
```

Example 3:
input:
```
2 2
4 2
3 3
```
output:
```
[(2,2),(4,3)]
```