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 the conditional construct (so no loops, arrays, functions, etc).

Problem 1: Printing Exams

Most courses have an exam at the end of the block to assess your knowledge. During an exam, you are given a physical copy of the exam. Printing all of these exams is a time-consuming task. In order to prevent waste, all exams are printed double-sided. In this problem you need to determine how many double-sided pages are needed to accommodate all students. Consider an exam with 8 single-sided pages. This exam can be printed on 4 double-sided pages. If 50 students attend the exam, a total of 200 pages need to be printed.

Write a program that reads from the input two integers n and m, where n is the number of single-sided pages and m is the number of students attending the exam. The program should output the total number of double-sided pages needed.

Example 1:

input:
```
8 50
```
output:
```
200
```

Example 2:

input:
```
3 50
```
output:
```
100
```

Example 3:

input:
```
2 2
```
output:
```
2
```

Problem 2: Product Divisors

In this exercise you need to determine whether a number is divisible by the product of the numbers' digits. One such number is for example 224. The product of its digits is $2 \cdot 2 \cdot 4 = 16$. Dividing 224 by 16 yields no remainder.

Write a program that reads from the input an integer n (where $1 \leq n < 10000$) and prints on the output `YES` if the n can be divided by the product of its digits without leaving a remainder. Otherwise print `NO` on the output. As stated earlier, you are **NOT** allowed to use loops in your solution.

Example 1:
input:
`224`
output:
`YES`

Example 2:
input:
`36`
output:
`YES`

Example 3:
input:
`37`
output:
`NO`

Problem 3: Odds & Evens

Integers consist of one or multiple digits. In this exercise you are asked to have a closer look at the digits of a number. Take for example the number `2247`. It consists of three even digits and one odd digit.

Write a program that reads from the input an integer n (where $1 \leq n < 10000$) and outputs `EVENS` if the number of even digits outnumber the number of odd digits. If the number of odd digits outnumber the number of even digits, then print `ODDS`. In case the number of odd and even digits are equal, you should print `BALANCED`. As stated earlier, you are **NOT** allowed to use loops in your solution.

Example 1:
input:
`2247`
output:
`EVENS`

Example 2:
input:
`123`
output:
`ODDS`

Example 3:
input:
`12`
output:
`BALANCED`

Problem 4: Overlapping Discs

In this exercise you need to determine whether a set of discs overlap. A disc is defined by its (x, y)-coordinates and its radius r. Two examples are listed in the figure below. In the first figure there are two overlapping discs and in the second figure there are three overlapping discs. A disc is not considered to be overlapping if it is only touched by another disc (see *example 1*).

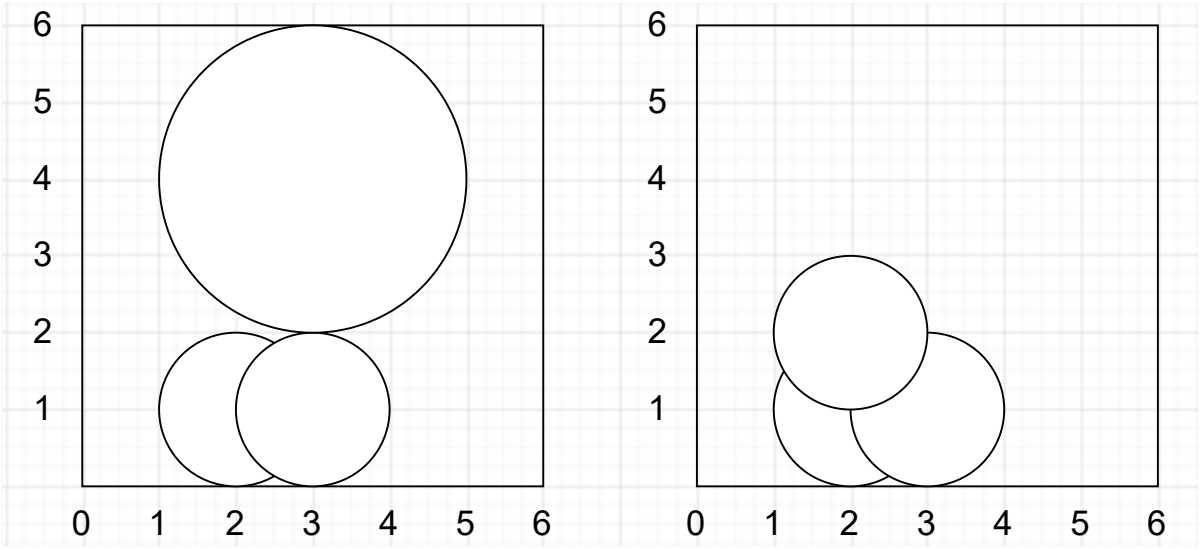

Figure 1: Two situations. Left: two overlapping discs. Right: three overlapping discs.

Write a program that reads from the input three lines. Each line holds the (x, y)-coordinate and radius r of a disc. Example inputs 1 and 2 correspond to the two examples depicted in the figure. The output is a single number representing the number of overlapping discs.

Example 1:

input:

```
2 1 1
3 1 1
3 4 2
```

output:

```
2
```

Example 2:

input:

```
2 1 1
3 1 1
2 2 1
```

output:

```
3
```

Example 3:

input:

```
2 1 1
5 1 1
3 4 1
```

output:

```
0
```