# Programming languages (TC-2006)

**Extra points challenge 08**

In this challenge, In this challenge you will practice what you know of the Erlang language by implementing some concurrent processes. Please consider that the purpose of this challenge is to allow you to practice and identify strengths and weaknesses regarding concurrency. Then, implementing any of the following functions without concurrency will invalidate your answer.

## 1 `increase` (25%)

Create a process that responds only to messages that are numbers. The process must respond by printing on screen the number it received increased by one. The process will remain alive as long as the message received contains only a number. When the message received is not a number, the process ends.

## 2 `dynamicIncrease` (25%)

Create a process that responds only to messages that are numbers. The first time the process receives a numeric mesage, it must respond by printing on screen the number it received increased by one. The second time, the number will be increased by two and so on. The process will remain alive as long as the message received contains only a number. When the message received is not a number, the process ends.

## 3 `calculator` (25%)

By using Erlang's concurrent processes, implement a calculator with internal memory. This calculator receives messages in the form `{Operation, Value}`, where `Operation` is one of the following atoms: `xadd`, `xsub`, `xmul` or `xdiv` (which correspond to mathematical operations), and `Value` is a number. When the calculator is initialized, its memory is set to one specific value indicated by the user. Then, all the operations conducted on the calculator are applied to the value of the internal memory and the one passed as part of the message (the value in the memory is always the first operand in the operation). The result of the operation is automatically saved into the internal memory of the calculator. When the calculator receives the atom `finish` it prints the value stored in the internal memory and ends.

## 4 `tictac` (25%)

By using Erlang's concurrent processes, implement a tic-tac clock. This timer will consist of to processes that comunicate with ech other. Each one of these processes will print a message on screen. So, one process prints "tic" and the other prints "tac" (yes, very creative, I know). The user will be able to start the clock by sending a message to any of the two processes. When the clock is started, messages "tic" and "tac" will be printed on screen, one after the other with a delay of one second between messages. In other words, the user will see "tic" on screen, and after one second it will see "tac", followed by another "tic" after another second, and so on. When the clock is stopped (a maximum number of tics must be specified by the user), the two processes must be stopped and then, no more messages will be shown on screen.

## Deliverables

SUBMIT Prepare an ERL file that contains the function requested and submit it to Canvas. **Please, do not submit other formats but ERL**.

**I promise to apply my knowledge, strive for its development, and not use unauthorized or illegal means to complete this activity, following the Tecnológico de Monterrey Student Code of Honor**.