Programming languages (TC-2006)

Homework 15

In this homework, you will use the Erlang language to implement some concurrent functions. Please note that solving these problems without using concurrency will not be considered for grading.

1 `append` (20%)

Implement a process in Erlang that listens to messages and, while the received message contains a positive number, the process adds such a number to the end of the list that contains all the positive numbers received so far. Whenever the process receives something that is not a positive number it stops and prints, on screen, the list with all the elements added so far.

2 `friend` (20%)

Implement a process in Erlang that listens to messages from other processes and, according to the messages the process receives, it decides to accept or ignore the message, based on the 'color' of the sender process. If both the sender and the receiver processes have the same 'color', then the processes are friends; and the message is 'accepted' (it prints the corresponding message on screen). If the message comes from a process which is not a friend, such a message is ignored.

Please note that for this exercise, none of the messages sent from the Erlang Shell contain information about the 'color' of the processes, as this is something handled by each process.

3 `marcopolo` (30%)

This time you will implement a concurrent version of the "Marco-Polo" game. The rules are simple:

- Player Polo (the one who hides) will be represented by one concurrent process.
- Player Marco (the one who seeks) will be represented by another concurrent process.
- You module must provide a function that initializes all the variables and starts the concurrent processes for the players. Both Marco and Polo should be randomly placed on a 20×20 squared board. You may find function `rand:uniform/1` useful for this.
- In the real-world game, Marco should shout "Marco" and Polo would answer "Polo", allowing Marco to change its position based on the acoustic. In this simulation, Marco will send a message to Polo with Marcos' current location (x and y coordinates) and Polo will send a message back to Marco with information that will allow Marco to change its position to get closer to Polo.
- When Marco finds Polo, both processes must finish.
- The game must print the proper messages on screen.

The following is an example run of the game (your messages may not necessarely be the same):

```
1> hw15Solution:test03().
Marco starts at position (17, 4)
Polo is hidden (we do not know where he is)...
Marco moves to (16, 5)
ok
Marco moves to (15, 6)
Marco moves to (14, 7)
Marco moves to (13, 8)
Marco moves to (12, 9)
Marco moves to (11, 10)
Marco moves to (11, 11)
Marco moves to (11, 12)
Marco moves to (11, 13)
Marco moves to (11, 14)
Marco moves to (11, 15)
Marco moves to (11, 16)
Marco moves to (11, 17)
Marco moves to (11, 18)
Marco found me! I was hiding at position (11, 18).
```

4 `bank` (30%)

By using Erlang's concurrent processes, implement a system that simulates a concurrent bank system. The system requires you to create a process that handles some requests and executes the proper actions. The operations allowed for this system are:

- Create account. Create a new account. The account identifier must be a number.
- Print balances. Prints the current balance of all the accounts.
- Deposit. Increase the amount of a particular account. The minimum amount to deposit is 20 USD.
- Withdraw. Decrease the amount in the account. The minimum amount to withdraw is 1 USD.

Please note that although the system is concurrent, we will not observe this behaviour since the test cases are sequential for the sake of evaluation. However, your system should be able to properly implemented by using processes and messages. Failing to do so will invalidate your answer.

Deliverables

Prepare an ERL file that contains the functions requested (in its corresponding module) and submit it to Canvas. **Please, do not submit other formats but ERL**. To prepare your ERL file, use the code template distributed along with this document. The template contains some test cases for each function to help you verify that your codes work as requested.

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.