Erlang – procesy

zadania

Zadanie 1

Napisać moduł `zadanie1`, który uruchomi N procesów, z których każdy wypisze "Proces " oraz swój numer, i każdy zrobi to 5 razy (czyli uogólnienie programu ze slajdów, w którym procesy były dwa). Liczbę N przekazujemy jako argument funkcji `start/1`.

Zadanie 2

Napisać moduł `zadanie2`, który uruchomi 2 procesy, A i B. Proces A ma wysłać atom czesc do procesu B, a proces B potwierdzić jego odbiór. Przykładowe wyjście:

```
Start procesu B, PID = <0.94.0>.
Start procesu A, PID = <0.95.0>.
Proces A wyslal atom czesc do B.
Proces B otrzymał atom czesc.
```

Zadanie 3

Napisać moduł `zadanie3`, który uruchomi 2 procesy, klient i serwer. Klient ma otrzymać jako argumenty listę liczb całkowitych oraz PID serwera i wysłać listę do serwera element po elemencie. Serwer każdą otrzymaną liczbę ma podnieść do kwadratu i wypisać. Po przesłaniu ostatniego elementu listy klient powinien przesłać atom koniec, po otrzymaniu którego serwer ma zakończyć działanie. Przykładowe wyjście:

```
Start serwera, PID = <0.87.0>.
Start klienta, PID = <0.88.0>, z listą [1,2,3].
1
4
9
Serwer konczy dzialanie.
```

Zadanie 4

Napisać moduł `zadanie4` analogiczny do modułu `zadanie3`, ale obliczenia powinny zostać przeprowadzone po stronie serwera (czyli dostaje liczbę, oblicza kwadrat, odsyła wynik, i to klient wypisuje wynik). Zamiast osobnego atomu `koniec`, serwer powinien sam się wyłączyć po 3 sekundach, w których nie otrzymał komunikatu.
Przykładowe wyjście:

```
Start serwera, PID = <0.95.0>.
Start klienta, PID = <0.96.0>, z listą [1,2,3].
Klient wyslal 1.
Klient dostal 1.
Klient wyslal 2.
Klient dostal 4.
Klient wyslal 3.
Klient dostal 9.
Serwer konczy dzialanie.
```

(ostatnia linijka po 3 sekundach)

Zadanie 5

Napisać moduł `zadanie5`, który posiada osobną funkcję startującą serwer i osobną funkcję startującą klienta. Należy też wykorzystać do ich komunikacji `register`. Serwer ma w nieskończonej pętli czekać na komunikaty, w zależności od tego, czy otrzyma liczbę, czy listę, ma wypisać stosowną wiadomość. Po otrzymaniu atomu `koniec` ma zakończyć działanie. W szczególności powinniśmy móc wystartować serwer jednym wywołaniem, a potem móc wywoływać wiele razy tworzenie klienta, za każdym razem z innym argumentem i serwer ma odpowiadać na komunikaty przesyłane w argumencie aż do otrzymania `koniec`.