Języki i paradygmaty programowania
Lista 4 (elementy języka Erlang)

Przemysław Kobylański + KNSI

Zaprogramuj w języku Erlang rozwiązania poniższych zadań. Na ocenę dostateczną trzeba rozwiązać wszystkie zadania bez gwiazdek. Na ocenę dobrą trzeba dodatkowo rozwiązać wszystkie zadania z jedną gwiazdką. Na ocenę bardzo dobrą trzeba rozwiązać wszystkie zadania.

Zadanie 1

Napisz funkcję `pythag(D)`, której wartością jest lista wszystkich trójek `{A, B, C}` liczb naturalnych dodatnich o sumie $A+B+C$ równej zadanemu D, które są trójką pitagorejską, tj. $A^2+B^2=C^2$.

Postaraj się jak efektywniej wyznaczać wszystkie takie trójki.

Przykład

```
1> c(pythag).
{ok, pythag}
2> pythag:pythag(100000).
[{20000,37500,42500},{21875,36000,42125}]
```

Zadanie 2*

Binarne drzewo uporządkowane względem kluczy i przechowujące w węzłach pary klucz-wartość będziemy zapisywać za pomocą następujących danych:

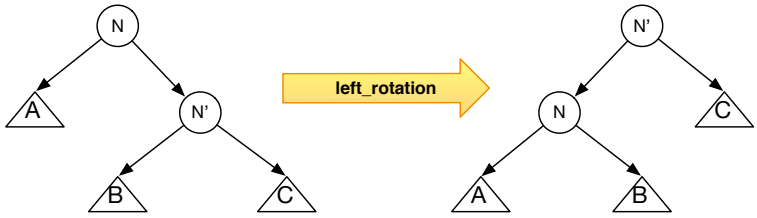
- stała `nil` reprezentuje drzewo puste,
- krotka `{node, Key, Value, Left, Right}` reprezentuje drzewo o parze `Key-Value` przechowywanej w korzeniu i dwóch poddrzewach: lewym `Left` i prawym `Right`.

Napisz funkcję `left_rotation(N)`, która dla węzła `N` wykonuje prawą rotację jak na rysunku 1.

Jeśli argumentem wywołania funkcji `left_rotation/1` nie jest odpowiedni węzeł drzewa binarnego, to funkcja powinna zwracać wartość `error`.

Przykład

```
1> c(rotation).
{ok,rotation}
2> rotation:left_rotation({node, k1, v1, nil, {node, k2, v2, nil, nil}}).
{node,k2,v2,{node,k1,v1,nil,nil},nil}
3> rotation:left_rotation({node, k1, v1, nil, nil}).
error
```



Rysunek 1: Lewa rotacja w węźle N

Zadanie 3**

Napisz moduł `przedszkole`, który eksportuje funkcję `przedszkolanka(N)`. Funkcja ta tworzy proces obsługujący pracę przedszkolanki mogącej opiekować się maksymalnie `N` dziećmi i jako wartość zwraca PID utworzonego procesu.

Proces przedszkolanki powinien obsługiwać następujące komunikaty wysyłane do niego:

`{Kto, pozostaw, ImięDziecka}` gdzie `Kto` jest PIDem nadawcy komunikatu a ImięDziecka jest łańcuchem znaków. Jeśli przedszkolanka może przyjąć kolejne dziecko (nie został przekroczony limit), to odsyłany jest komunikat `{ok, ImięDziecka}`, natomiast w przeciwnym przypadku odsyłany jest komunikat `error`.

`{Kto, odbierz, ImięDziecka}` gdzie `Kto` jest PIDem nadawcy komunikatu a ImięDziecka jest łańcuchem znaków. Jeśli proces o PIDzie `Kto` pozostawił dziecko o imieniu `ImięDziecka`, to odsyłany jest komunikat `{ok, ImięDziecka}`, natomiast w przeciwnym przypadku odsyłany jest komunikat `error`.

Jak Twój program zareaguje gdy pozostawi się przedszkolance dwójkę dzieci o tym samym imieniu? W jaki sposób przechowujesz oddane pod opiekę przedszkolanki dzieci? W jaki sposób testujesz poprawność odsyłanych do rodzica komunikatów? Czy użyłeś zaprezentowaną na wykładzie funkcję RPC?

Literatura

[1] J. Armstrong. Programming Erlang. Software for a Concurrent World. Pragmatic Bookshelf, 2007.