1. Для этого задания используйте данные из 2-й ДР переменной Persons.

   Выполните команды:

   1. *[_, SecondPerson, _, _] = Persons.*

      ```
      3> [_, SecondPerson, _, _] = Persons.
      ```

      Проверьте и объясните присвоенное значение *SecondPerson*.

      ```
      4> SecondPerson.
      #person{id = 2,name = "Kate",age = 20,gender = female}
      ```

      Связали переменную SecondPerson со списком Persons.
   2. *SecondName = SecondPerson#person.name.*

      ```
      5> SecondName = SecondPerson#person.name.
      "Kate"
      ```

      SecondAge = SecondPerson#person.age.

      ```
      6> SecondAge = SecondPerson#person.age.
      20
      ```

      Через rd вывели имя и возраст второго человека из списка.
   3. *[_, #person{name = SecondName, age = SecondAge} | _Rest] = Persons.*
      SecondName.

      ```
      8> SecondName.
      "Kate"
      ```

      SecondAge.

      ```
      9> SecondAge.
      20
      ```

      Связали имя с SecondName и возраст с SecondAge.
      Проверьте присвоенные *на 2 шаге* значения *SecondName*, *SecondAge* и объясните почему на 3 шаге сопоставление с образцом (pattern matching) прошло успешно.
   4. *Persons.*

      ```
      10> Persons.
      [#person{id = 1,name = "Bob",age = 23,gender = male},
       #person{id = 2,name = "Kate",age = 20,gender = female},
       #person{id = 3,name = "Jack",age = 34,gender = male},
       #person{id = 4,name = "Nata",age = 54,gender = female}]
      ```

      Проверьте, что список Persons не изменился.
   5. *SecondPerson#person{age = 21}.*

      ```
      11> SecondPerson#person{age = 21}.
      #person{id = 2,name = "Kate",age = 21,gender = female}
      ```

      Проверьте, что список Persons и *SecondPerson* не изменились.

      ```
      12> Persons.
      [#person{id = 1,name = "Bob",age = 23,gender = male},
       #person{id = 2,name = "Kate",age = 20,gender = female},
       #person{id = 3,name = "Jack",age = 34,gender = male},
       #person{id = 4,name = "Nata",age = 54,gender = female}]
      ```

2. Создайте список maps, которые содержат данные о 4х человек. Данные возьмите те же самые, что мы использовали до этого в ДР 2.
Присвойте данные переменной Persons.
Persons = [#{id => 1, name => "Karl", age => 87, gender => male},
*#{id => 2, name => "*Kate*", age => 86, gender => female}, …]*

Выполните команды:

1. *[FirstPerson | _] = Persons.*

```
3> [FirstPerson | _] = Persons.
[#{id => 1,name => "Bob",age => 23,gender => male},
 #{id => 2,name => "Kate",age => 20,gender => female},
 #{id => 3,name => "Jack",age => 34,gender => male},
 #{id => 4,name => "Nata",age => 54,gender => female}]
4> FirstPerson.
#{id => 1,name => "Bob",age => 23,gender => male}
```

   Проверьте и объясните присвоенное значение *FirstPerson*.
   Связали FirstPerson со списком Persons.
2. *[_, _, #{name := Name, age := Age}, _] = Persons.*

```
5> [_, _, #{name := Name, age := Age},  _] = Persons.
[#{id => 1,name => "Bob",age => 23,gender => male},
 #{id => 2,name => "Kate",age => 20,gender => female},
 #{id => 3,name => "Jack",age => 34,gender => male},
 #{id => 4,name => "Nata",age => 54,gender => female}]
```

   Проверьте и объясните присвоенное значение *Name, Age*.
   Name.

```
6> Name.
"Jack"
```

   Age.

```
7> Age.
34
```

3. *[_First, _Second, #{name := Name, age := Age} | _Rest] = Persons.*

```
8> [_First, _Second, #{name := Name, age := Age} | _Rest] = Persons.
[#{id => 1,name => "Bob",age => 23,gender => male},
 #{id => 2,name => "Kate",age => 20,gender => female},
 #{id => 3,name => "Jack",age => 34,gender => male},
 #{id => 4,name => "Nata",age => 54,gender => female}]
```

   Проверьте и объясните присвоенные значения Name, Age в пункте 2 и 3.
   Почему команда 3 завершилась успешно (Name, Age уже связаны)?
4. *Persons.*

```
11> Persons.
[#{id => 1,name => "Bob",age => 23,gender => male},
 #{id => 2,name => "Kate",age => 20,gender => female},
 #{id => 3,name => "Jack",age => 34,gender => male},
 #{id => 4,name => "Nata",age => 54,gender => female}]
```

   Проверьте, что список Persons не изменился.
5. *FirstPerson#{age := 24}.*

```
12> FirstPerson#{age := 24}.
#{id => 1,name => "Bob",age => 24,gender => male}
```

Проверьте, что список Persons и *FirstPerson* не изменились. Почему?

6. *FirstPerson#{address := “Mira 31”}.*

```
#{id => 1,name => "Bob",age => 24,gender => male}
13> FirstPerson#{address := “Mira 31”}.
Mira 31”}.
* 1:25: illegal character
```

Проверьте и объясните результат.

В rd мы не связывали addres.

3. Создайте модуль *converter.erl* с функцией to_rub/1 на вход которой поступает тип валюты и сумма и возвращается результат конвертации в рубли {ok, Result} или {error, badarg}.

Пример:

```
to_rub({usd, Amount}) ->
        io:format(“Convert ~p to rub, amount ~p~n”, [usd, Amount]),
         {ok, Amount * 75.5};
to_rub({euro, Amount}) ->
        io:format(“Convert ~p to rub, amount ~p~n”, [euro, Amount]),
         {ok, Amount * 80};
        …
```

converter.erl

```erlang
-module(converter).

-export([to_rub/1]).

to_rub({usd, Amount}) when is_integer(Amount), Amount > 0 ->
    io:format("Convert ~p to rub, amount ~p~n", [usd, Amount]),
    {ok, Amount*75.5};
to_rub({euro, Amount}) when is_integer(Amount), Amount > 0 ->
    io:format("Convert ~p to rub, amount ~p~n", [euro, Amount]),
    {ok, Amount*80};
to_rub({lari, Amount}) when is_integer(Amount), Amount > 0 ->
    io:format("Convert ~p to rub, amount ~p~n", [lari, Amount]),
    {ok, Amount*29};
to_rub({peso, Amount}) when is_integer(Amount), Amount > 0 ->
    io:format("Convert ~p to rub, amount ~p~n", [peso, Amount]),
    {ok, Amount*3};
to_rub({krone, Amount}) when is_integer(Amount), Amount > 0 ->
    io:format("Convert ~p to rub, amount ~p~n", [krone, Amount]),
    {ok, Amount*10};
to_rub(Error) ->
    io:format("Error convert ~p to rub~n", [element(1, Error)]),
    {error, badarg}.
```

5 валют и курс конвертации в рубли:
Usd - 75.5; Euro - 80; Lari - 29; Peso - 3; Krone - 10
Для всех остальных валют верните ошибку {error, badarg} и выведите ошибку на экран с помощью функции io:format/2.
Используйте проверку что Amount целое число и больше нуля.

Запустите EShell из папки где у вас находится файл *converter.erl*
В консоли скомпилируйте файл и вызовите функции:
c(“converter.erl”).
converter:to_rub({usd, 100}).
converter:to_rub({peso, 12}).

converter:to_rub({yene, 30}) – Валюта Йена отсутствует, возвращает ошибку.
converter:to_rub({euro, -15}) – отрицательное значение.

Прокомментируйте строки в которых вызов функции завершился ошибкой.

```
Eshell V14.1 (press Ctrl+G to abort, type help(). for help)
1> c(converter.erl).
* 1:12: syntax error before: '.'
1> c("converter.erl").
{ok,converter}
2> converter:to_rub({usd, 100}).
Convert usd to rub, amount 100
{ok,7550.0}
3> converter:to_rub({peso, 12}).
Convert peso to rub, amount 12
{ok,36}
4> converter:to_rub({yene, 30}).
Error convert yene to rub
{error,badarg}
5> converter:to_rub({euro, -15}).
Error convert euro to rub
{error,badarg}
```