The link of the [documentation.](http://erlang.org/doc/man/)

--------------------------------------------------------

Task no. 0:

===========

Copy the following declaration form to your Erlang file and fill in the required data:

~~~~

%% <Name>

%% <Neptun ID>

%% <DDS, TEST1>

%% <22.03.2018.>

%% This solution was submitted and prepared by <Name, Neptun ID> for the DDS Retake Test Sequential.

%% I declare that this solution is my own work.

%% I have not copied or used third party solutions.

%% I have not passed my solution to my classmates, neither made it public.

%% Students’ regulation of Eötvös Loránd University (ELTE Regulations Vol. II. 74/C. § ) states that as long as a student presents another student’s work - or at least the significant part of it - as his/her own performance, it will count as a disciplinary fault. The most serious consequence of a disciplinary fault can be dismissal of the student from the University.

~~~~

Task no. 1:

=================================

Ethiopian multiplication is a method of multiplying integers using only addition, doubling, and halving.

Define three functions:

one to halve an integer,

one to double an integer, and

one to state if an integer is even.

Use them to create multiplier(L,R), that takes the two numbers (Left and Rigth) and compute their product in the following way:

Half the Left number until you get to 1, at the same time Double the Rigth number for the same amount of times.

Discard any value from Rigth where the corresponding value in Left is even.

Sum all number left in right to produce the result of L\*R.

Ex. 17 34 discard 68,136 and 272

8 68 34 + 544 == 17\*34

4 136

2 272

1 544

%%%%%%%%%%%%#1

~~~

multiplier(integer(),integer()) -> integer()

~~~

Test cases:

-----------

\*\*Do not forget to change the name of the module!\*\*

~~~

test:multiplier(12,3)==(12\*3).

test:multiplier(0,0)== forbidden.

test:multiplier(7,0)== forbidden.

~~~

Task no. 2: is\_numeric/1

==========================

Create the function is\_numeric(S), that check if a given string is composed only by numbers (Integer and Float is good), return true if the argument passed is either an integer or a float.

\*\*IMPORTANT:\*\* you need to reimplement the function, do not use the built-in function is\_number/1.

~~~

is\_numeric(S:list()) -> boolean()

~~~

Test cases:

-----------

\*\*Do not forget to change the name of the module!\*\*

~~~

test:is\_numeric("123") ==true.

test:is\_numeric("12.3")==true.

test:is\_numeric(".12.3")==false.

test:is\_numeric(".12.3.")==false.

test:is\_numeric("123.")==false.

test:is\_numeric("")==false.

~~~

Task no. 3: replace/3

==============================

Define the replace/3 function that takes three strings as arguments and

replaces every other occurrence of the substring Old with New in Str.

~~~

replace(Str::list(), Old::list(), New::list())-> list()

~~~

Test cases:

-----------

\*\*Do not forget to change the name of the module!\*\*

~~~

test:replace("AppleAppleApple", "Apple", "Pear")=="PearApplePear"

test:replace("AppleAppleApple", "Apple", "")=="Apple"

test:replace("AppleAppleApple", "App", "Pear")=="PearleApplePearle"

~~~

Task no. 4: partition/2

=================

Define the higher-order function partition/2, which will take a list and a boolean Predicate P and return two Lists: the first one that has the terms which satisfy P, and the second one a list for the others terms.

\*\*IMPORTANT:\*\* Use recursion, you are not allowed to use the built in function lists:partition/2.

~~~

partition(Predicate::fun(any() -> boolean()), List(E))->Result::list(E)

~~~

test:partition(fun(X)-> X>10 end ,[7,10,8,155,133] )==[[155,133],[7,10,8]]

test:partition(fun(X)-> X rem 2==0 end ,[7,8,2222,223] )==[[8,2222],[7,223]]

test:partition(fun(X)-> is\_atom(X) end ,[[1,2],[alma],hello] )==[[hello],[[1,2],[alma]]]

~~~