```
-module(sendmory).
-author('patai@iit.bme.hu, hanak@inf.bme.hu').
-vsn('2010-11-27').
-compile(export_all).

%      S E N D
%    + M O R E
%  = M O N E Y

% @type d() = integer(). % digit values have to be in the range 0..9.
% @type octet() = {d(),d(),d(),d(),d(),d(),d(),d()}.

% @spec num(Ns::[d()]) -> N::integer().
% The value of the digit list Ns, interpreted as a decimal number, is N.
num(Ns)->
    lists:foldl(fun(X,E) -> E*10+X end, 0, Ns).


% @spec smm0() -> [octet()].
% All the checks are after the generators (generate and test).
smm0() ->
    Ds = lists:seq(0, 9),
    [{S,E,N,D,M,O,R,Y} ||
        S <- Ds, E <- Ds, N <- Ds, D <- Ds,
        M <- Ds, O <- Ds, R <- Ds, Y <- Ds,
        all_different([S,E,N,D,M,O,R,Y]),
        S > 0, M > 0,
        begin Send = num([S,E,N,D]),
              More = num([M,O,R,E]),
              Money = num([M,O,N,E,Y]),
              Send+More == Money
        end].


% @spec all_different(Xs::[any()]) -> B::bool()
% B is true, if there is no repeated value in the list Xs.
all_different([]) ->
    true;
all_different([X|Xs]) ->
    not lists:member(X,Xs) andalso all_different(Xs).

% all_different(L) -> length(L) == length(lists:usort(L)).


% @spec smm1() -> [octet()].
% Checking inequalities on the way.
smm1() ->
    Ds = lists:seq(0, 9),
    [{S,E,N,D,M,O,R,Y} ||
        S <- Ds,
        E <- Ds, E /= S,
        N <- Ds, not lists:member(N, [S,E]),
        D <- Ds, not lists:member(D, [S,E,N]),
        M <- Ds, not lists:member(M, [S,E,N,D]),
        O <- Ds, not lists:member(O, [S,E,N,D,M]),
        R <- Ds, not lists:member(R, [S,E,N,D,M,O]),
        Y <- Ds, not lists:member(Y, [S,E,N,D,M,O,R]),
        S > 0, M > 0,
        begin Send = num([S,E,N,D]),
              More = num([M,O,R,E]),
              Money = num([M,O,N,E,Y]),
              Send+More == Money
        end].
```

```
% @spec perm() -> [octet()].
% Generating permutations while excluding digits that are already in use.
perm() ->
    Ns = lists:seq(0,9),
    [{S,E,N,D,M,O,R,Y} ||
        S <- Ns -- [0], % Zero (0) excluded.
        E <- Ns -- [S],
        N <- Ns -- [S,E],
        D <- Ns -- [S,E,N],
        M <- Ns -- [0,S,E,N,D], % Zero (0) excluded.
        O <- Ns -- [S,E,N,D,M],
        R <- Ns -- [S,E,N,D,M,O],
        Y <- Ns -- [S,E,N,D,M,O,R]
    ].


% @spec check(octet()) -> bool().
% Checking the rule.
check({S,E,N,D,M,O,R,Y}) ->
    (1000*S + 100*E + 10*N + D) + (1000*M + 100*O + 10*R + E) ==
        (10000*M + 1000*O + 100*N + 10*E + Y).


% @spec smm2() -> [octet()].
% First generate, then test.
smm2() ->
    lists:filter(fun check/1, perm()).


% @spec smm3() -> [octet()].
% First generate, then test.
smm3() ->
    Ns = lists:seq(0,9),
    [{S,E,N,D,M,O,R,Y} ||
        S <- Ns -- [0],
        E <- Ns -- [S],
        N <- Ns -- [S,E],
        D <- Ns -- [S,E,N],
        M <- Ns -- [0,S,E,N,D],
        O <- Ns -- [S,E,N,D,M],
        R <- Ns -- [S,E,N,D,M,O],
        Y <- Ns -- [S,E,N,D,M,O,R],
        (1000*S + 100*E + 10*N + D) + (1000*M + 100*O + 10*R + E) ==
            (10000*M + 1000*O + 100*N + 10*E + Y)
      ].


% @spec smm4() -> [octet()].
% Handling inequalities by construction (which is actually slower...).
smm4() ->
    [{S,E,N,D,M,O,R,Y} ||
        {[S,E,N,D,M,O,R,Y],_} <- select_n(8, lists:seq(0, 9)),
        S > 0, M > 0,
        begin Send = num([S,E,N,D]),
              More = num([M,O,R,E]),
              Money = num([M,O,N,E,Y]),
              Send+More == Money
        end].
```

```erlang
%% The below version of select_n may be used with smm4/0 since smm4/0
%% doesn’t have to return the remaining elements.
% @spec select_n(N::integer(),Xs::[any()] -> Yss::[[any()]].
% Elements of Yss are Ys::[any()] lists of length N, where each Ys
% contains all possible permutations of N distinct elements of Xs.
% Fails miserably unless (N > 0 andalso N =< length(Xs)) holds;
%%% select_n(N, Xs) ->
%%%     [[Y|Ys] ||
%%%     {Y,More} <- select(Xs),
%%%     Ys <- case N of
%%%               1 -> [[]];
%%%               _ -> select_n(N-1, More)
%%%           end].


% @spec select_n(N::integer(),Xs::[any()] -> Zs::[{[any()],[any()]}].
% Elements of Zs are {Ys::[any()],Rs::[any()]} pairs, where Ys, a list of
% length N, contains all possible permutations of N distinct elements of Xs,
% and Rs contains all other elements of Xs not contained in Ys.
% Fails miserably unless (N > 0 andalso N =< length(Xs)) holds;
% it returns remaining elements for later use.
select_n(N, Xs) ->
    [{[Y|Ys],Rest} ||
        {Y,More} <- select(Xs),
        {Ys,Rest} <- case N of
                         1 -> [{[],More}];
                         _ -> select_n(N-1, More)
                     end].


% @spec select(Xs::[any()] -> Zs::[{any(),[any()]}].
% Elements of Zs are {X::any(),Rs::[any()]} pairs, where each X is a distinct
% element of Xs and Rs contains all other elements of Xs different from X,
% while length(Zs) == length(Xs), i.e. all elements of Xs occur as X in Zs.
select([X]) ->
    [{X,[]}];
select([X|Xs]) ->
    [{X,Xs}|[{Y,[X|Ys]} || {Y,Ys} <- select(Xs)]].


% @spec smm5() -> [octet()].
% Building from right to left, checking the partial sums.
smm5() ->
    Ds0 = lists:seq(0, 9),
    [{S,E,N,D,M,O,R,Y} ||
        {[D,E,Y],Ds1} <- select_n(3, Ds0),
        (D+E) rem 10 == Y,
        {[R,N],Ds2} <- select_n(2, Ds1),
        (num([N,D])+num([R,E])) rem 100 == num([E,Y]),
        {[O],Ds3} <- select_n(1, Ds2),
        (num([E,N,D])+num([O,R,E])) rem 1000 == num([N,E,Y]),
        {[S,M],_} <- select_n(2, Ds3),
        S > 0, M > 0,
        begin Send = num([S,E,N,D]),
              More = num([M,O,R,E]),
              Money = num([M,O,N,E,Y]),
              Send+More == Money
        end].
```

```erlang
% exercise: generalise the task to specifications given in strings:
%
% smm_gen("send","more","money")  ==>
% [{9567,1085,10652}]%
% smm_gen("four","five","nine")  ==>
% [{2970,2381,5351},{2970,2481,5451},...,{1970,1568,3538}]


% @type selector() -> integer().
% @spec stopper(selector()) -> ().
% Run the selected smm function and measure its run-time.
stopper(V) ->
    _T = statistics(runtime),
    [{S,E,N,D,M,O,R,Y}|_] =
        case V of
            0 ->
                smm0();
            1 ->
                smm1();
            2 ->
                smm2();
            3 ->
                smm3();
            4 ->
                smm4();
            5 ->
                smm5()
        end,
    {_,T} = statistics(runtime),
    io:format("SEND+MORE=MONEY: ~w~w~w~w+~w~w~w~w=~w~w~w~w~w~8wms~n",
              [S,E,N,D,M,O,R,E,M,O,N,E,Y,T]).
```