```


% DP, small excercises
% See also the document "Számolótábla" (dprSzamolotabla.pdf)

-module(dprgy1kf).
-compile(export_all).

%% Possibility for abbreviating a module name
%% D = dprgy1kf.
%% D:insert_...

% 1/1 Insert/delete an element at position N

%% Note: the guard is needed to detect negative arguments.

insert_nth_1(1, Y, Xs) ->
    [Y|Xs];
insert_nth_1(N, Y, [X|Xs]) when N > 1 ->
    [X|insert_nth_1(N-1, Y, Xs)];
insert_nth_1(_, _, _) ->
    error.

insert_nth_2(N, Y, Xs) ->
    case (N < length(Xs)+1) and (N > 0) of
        true ->
            lists:sublist(Xs, N-1)++[Y|lists:nthtail(N-1, Xs)];
        _ ->
            error
    end.

% 1/2, 1/3 Key-value assignments using a list of pairs,
% looking up for a key

%% Note: although no key can be repeated in a well-formed dictionary the
%% pattern [V|_] is used in find_key_3 to ensure that it behaves identical
%% to find_key_1 and find_key_2 also in cases when a key occurs repeatedly.

find_key_1(K, [{K,V}|_]) ->
    V;
find_key_1(K, [_|KVs]) ->
    find_key_1(K, KVs);
find_key_1(_, []) ->
    not_found.

find_key_2(K, KVs) ->
    case lists:keysearch(K, 1, KVs) of
        {value,{_,V}} ->
            V;
        false ->
            not_found
    end.

find_key_3(K, KVs) ->
    case [EV || {EK,EV} <- KVs, EK == K] of
        [V|_] ->
            V;
        [] ->
            not_found
    end.

```

```erlang


% 1/4 Split a list into two parts by providing the length of the first part

%% Note: the guard is needed to detect negative arguments.

slice_list_1(0, Xs) ->
    {[],Xs};
slice_list_1(N, [X|Xs]) when N > 0 ->
    case slice_list_1(N-1, Xs) of
        {L1,L2} ->
             {[X|L1],L2};
        _ ->
            error
    end;
slice_list_1(_, _) ->
    error.

%% Note: slice_list_2 (lists:split) raises an exeption if N is invalid.

slice_list_2(N, Xs) ->
    lists:split(N, Xs).

% 1/5 Zip two lists, unzip a list into two

zip_lists_1([X|Xs], [Y|Ys]) ->
    [{X,Y}|zip_lists_1(Xs, Ys)];
zip_lists_1([], _) ->
    [];
zip_lists_1(_, []) ->
    [].

zip_lists_2(Xs, Ys) ->
    lists:zip(Xs, Ys).

%% Note: the results of zip_lists_1 and zip_lists_2 (lists:zip) differ: the
%% first one truncates the longer list, if exists, while the second raises
%% an exception.

unzip_lists_1([{X,Y}|XYs]) ->
    {Xs,Ys} = unzip_lists_1(XYs),
    {[X|Xs],[Y|Ys]};
unzip_lists_1([]) ->
    {[],[]}.

unzip_lists_2(XYs) ->
    lists:unzip(XYs).

% 2/1 Returning a cell, row, column or block

read_cell_xy(X, Y, M) ->
    lists:nth(X, lists:nth(Y, M)).

read_row_y(Y, M) ->
    lists:nth(Y, M).

read_column_x(X, M) ->
    [lists:nth(X, L) || L <- M].

read_block(X, Y, XSize, YSize, M) ->
    [lists:sublist(L, X, XSize) || L <- lists:sublist(M, Y, YSize)].

```

```
% 2/4 Copying a slice or block

copy_block(X1, Y1, X2, Y2, XSize, YSize, M) ->
    Block = read_block(X1, Y1, XSize, YSize, M),
    % in Mid, cells from X2 to X2+XSize are replaced in each row
    Mid = [copy_slice(X2, XSize, L, S) ||
              % each row of M from Y2 to Y2+YSize is paired with the
              % corresponding row of Block
              {L,S} <- lists:zip(lists:sublist(M, Y2, YSize), Block)],
    % cells from row Y2 to Y2+Y2Size will be replaced by the content of Mid
    copy_slice(Y2, YSize, M, Mid).

demo_block(X1, Y1, _X2, Y2, XSize, YSize, M) ->
    Block = read_block(X1, Y1, XSize, YSize, M),
    [{L,S} ||
        % each row of M from Y2 to Y2+YSize is paired with the
        % corresponding row of Block
        {L,S} <- lists:zip(lists:sublist(M, Y2, YSize), Block)].

demo_bl(1) ->
    demo_block(2, 1, 1, 2, 2, 3, matrix(2)).

%% [{[2,4,6,8,10],[2,3]},
%%  {[3,6,9,12,15],[4,6]},
%%  {[4,8,12,16,20],[6,9]}]

% copy_slice(N, NSize, L, S): copy of list L where the sublist from N to
% N+NSize is replaced by list S
copy_slice(N, NSize, L, S) ->
    lists:sublist(L, N-1) ++ S ++ lists:nthtail(N+NSize-1, L).

% 2/3 Deleting a slice or block

clear_block(X, Y, XSize, YSize, M) ->
    Mid = [clear_slice(X, XSize, L) || L <- lists:sublist(M, Y, YSize)],
    copy_slice(Y, YSize, M, Mid).

clear_slice(N, NSize, L) ->
    copy_slice(N, NSize, L, lists:duplicate(NSize, empty)).

% Testlists, testmatrices

list(1) ->
    [a,b,c,d,e,f,g];
list(2) ->
    [1,2,3,4,5,6,7];
list(3) ->
    [{1,a},{2,b},{3,c},{4,d}];
list(4) ->
    [{1,a},{2,b},{3,c},{4,d},{3,d}].

matrix(1) ->
    [[a,b,c,d],[e,f,g,h],[i,j,k,l]];
matrix(2) ->
    [[X*Y || X <- lists:seq(1,5)] || Y <- lists:seq(1,5)].
```