```
function x_square = GenerateSquarePulses(t_axis,T_sq,E_bit,fs,x_bits,type)
%
% Inputs:
%   t_axis:     Time axis
%   T_sq_dur:   Duration of the square pulse in seconds
%   E_bit:      Total energy in all samples of one square pulse
%   fs:         Sampling frequency
%   x_bits:     Sequence of input bits (if not available, then it is equal
%               to 1)
%   type:       Type of bit coding, 'RZ' or 'NRZ' (default is 'RZ')
% Outputs:
%   x_square:   The sequence of samples corresponding to the pulse shaping
%               of the input bits using a square pulse shape
%
% This function takes an input a sequence of bits, x_bits. It then
% generates a sequence of samples in x_square, corresponding to the pulse
% shaping of these bits using square pulses. The parameters of the square
% pulse used for pulse shaping are given in the inputs of the function.
%
% Notes:
%       If x_bits is not specified, it is assumed to be equal to 1
%       bit.
%       If type is not specified, it is assumed to be 'RZ'.
%       x_square is always equal in dimention to t_axis.

if nargin < 5
    x_bits = 1;
    type = 'unipolar';
end

if nargin < 6
    type = 'unipolar';
end

Ts = 1/fs;
N = length(t_axis);

%%% Generate one square pulse
N_sq = round(T_sq/Ts);
one_square = zeros(1,N_sq);

%%% WRITE YOUR CODE HERE
% Here you should create exactly one square pulse with the specified
% parameters. This square pulse should be stored in the array called
% one_square. The dimensions of this array should be 1 x N_sq_pos. The
% length of the square pulse you generate should be equal to N_sq_pos,
% i.e., the variable one_square should not change dimensions after you
% generate the pulse. Keep in mind that you have to set the energy of the
% square pulse to be equal to E_bit.
x_square = zeros(1, N);
% YOUR CODE ENDS HERE
%%%

%%% Generate one square pulse for each bit in the variable x_bit
% Here, you should be able to use the pulse you generated in one_square to
% create the final array x_square. This final array should consist of the
% square pulses corresponding to each input bit. Note that the dimensions
% of the x_square should always be equal to the dimensions of t_axis.

switch type
    case ('bipolar')
        %%% This case is for NRZ
        %%% WRITE YOUR CODE HERE
        amp_bi = sqrt(E_bit/N_sq);
```

```
        for i=1:length(x_bits)
            if x_bits(i) == 1
                x_square((((N_sq-1)*(i-1)) + i): (i*N_sq)) = amp_bi + one_square;
            elseif x_bits(i) == 0
                x_square((((N_sq-1)*(i-1)) + i): (i*N_sq)) = -amp_bi + one_square;
            end
        end
        % YOUR CODE ENDS HERE

    case ('unipolar')
        %%% This case is for RZ
        %%% WRITE YOUR CODE HERE
        amp_uni = sqrt((2*E_bit)/N_sq);

        for i=1:length(x_bits)
            if x_bits(i) == 1
                x_square((((N_sq-1)*(i-1)) + i): (i*N_sq)) = amp_uni + one_square;
            end
        end

        % YOUR CODE ENDS HERE
end
```