```
function  [hh, H, W] = dtmfdesign(fcent, L, fs)
%DTMFDESIGN
%     hh = dtmfdesign(fcent, L, fs)
%       returns a matrix where each column is the
%       impulse response of a BPF, one for each frequency
%  fcent = vector of center frequencies
%      L = length of FIR bandpass filters
%     fs = sampling freq
%
% The BPFs must be scaled so that the maximum magnitude
% of the frequency response is equal to one.
%===========================================
% [697;770;852;941;1209;1336;1477;1633]; list of centre frequencies

%%%% add your lines below to complete the code

%w: normalised angular frequency
%h: frequency response vector


nn = 0:L;

for i=1:size(fcent)

    %calculate filter coeffs
    bb = cos(2*pi*(fcent(i)/fs)*nn);

    %get the greatest unscaled val to scale BPF down
    [h, w] = freqz(bb, 1, 4096);
    maxVal = max(abs(h)); %find the maximum value

    %scaled
    bb_scaled = (1/maxVal)*bb;
    [h, w] = freqz(bb_scaled, 1, 4096);
    hh(:, i) = bb_scaled;
    H(:, i) = h;
    W(:, i) = w;

    % %plot stuff
    % plot(w, abs(h));
    % xlabel('Normalised Frequency')
    % ylabel('Frequency Response Magnitude')
    % xlim([0, 3.14]);
    % ylim([-0.2, 1.2]);
    % hold on;
    %
    %
    % %find the cutoff frequencies
    % lcf_index = find(diff(abs(h) > 0.7071) == 1) + 1;
```

```
    % ucf_index = find(diff(abs(h) < 0.7071) == 1) + 1;
    % lcf = w(lcf_index);
    % ucf = w(ucf_index);
    %
    % fprintf('For frequency number %d\n', i)
    % disp('Lower cutoff frequency:');
    % disp((lcf*fs)/(2*pi));
    % disp('Upper cutoff frequency:');
    % disp((ucf*fs)/(2*pi));
    % disp('Passband width');
    % disp(((ucf-lcf)*fs/(2*pi)));

end
end
```