```
% Audio and Speech Signal Analysis in Time and Frequency Domains

% Load the audio signal
[audioSignal, fs] = audioread('mic_F01_sa2.wav');  % Replace with your audio file
audioSignal = audioSignal(:, 1);  % Use the first channel if stereo

% Time vector for plotting
t = (0:length(audioSignal)-1) / fs;

% Plot the waveform in time domain
figure;
plot(t, audioSignal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Time Domain Representation of the Audio Signal');
grid on;

% Compute and plot the frequency spectrum using FFT
n = length(audioSignal);
f = (0:n-1)*(fs/n);  % Frequency axis
Y = fft(audioSignal);

% Plot the single-sided amplitude spectrum
figure;
plot(f(1:n/2), abs(Y(1:n/2)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Spectrum of the Audio Signal');
grid on;

% Spectrogram analysis
windowSize = 256; % Define window size
overlap = 200;    % Define overlap samples
nfft = 512;       % Number of FFT points

figure;
spectrogram(audioSignal, windowSize, overlap, nfft, fs, 'yaxis');
title('Spectrogram of the Audio Signal');

% Play the original audio
disp('Playing the original audio...');
sound(audioSignal, fs);
pause(length(audioSignal)/fs + 1);

% Compute and display basic signal properties
signalDuration = length(audioSignal) / fs;
signalPower = rms(audioSignal)^2;

disp(['Signal Duration: ', num2str(signalDuration), ' seconds']);
disp(['Signal Power: ', num2str(signalPower), ' Watts']);

% Apply a band-pass filter to remove noise (example: 300Hz - 3400Hz for speech)
lowCut = 300;
highCut = 3400;
[b, a] = butter(4, [lowCut, highCut] / (fs / 2), 'bandpass');
```

```
filteredSignal = filtfilt(b, a, audioSignal);

% Plot filtered signal in time domain
figure;
plot(t, filteredSignal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Filtered Speech Signal (300Hz - 3400Hz)');
grid on;

% Play filtered audio
disp('Playing the filtered audio...');
sound(filteredSignal, fs);
```

2 Perform basic signal processing operations such as filtering and resampling

```
% Clear workspace and close figures
clear;
clc;
close all;

% Load the audio signal
[audioSignal, fs] = audioread('speech.wav');  % Replace with your audio file
audioSignal = audioSignal(:, 1);  % Use the first channel if stereo

% Display original sampling rate
disp(['Original Sampling Rate: ', num2str(fs), ' Hz']);

% Plot the original audio signal (time domain)
t = (0:length(audioSignal)-1) / fs;
figure;
subplot(2,1,1);
plot(t, audioSignal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Original Audio Signal');
grid on;

% Perform Low-Pass Filtering (to remove high-frequency noise)
fc = 3000;  % Cut-off frequency (Hz)
order = 6;  % Filter order
[b, a] = butter(order, fc/(fs/2), 'low');  % Design Butterworth low-pass filter
filteredSignal = filter(b, a, audioSignal);

% Plot the filtered audio signal
subplot(2,1,2);
plot(t, filteredSignal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Filtered Audio Signal (Low-pass at 3 kHz)');
grid on;

% Compute and plot frequency spectrum before and after filtering
n = length(audioSignal);
f = (0:n-1)*(fs/n);  % Frequency axis
```

```
Y_original = abs(fft(audioSignal));
Y_filtered = abs(fft(filteredSignal));

figure;
subplot(2,1,1);
plot(f(1:n/2), Y_original(1:n/2));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Spectrum of Original Signal');
grid on;

subplot(2,1,2);
plot(f(1:n/2), Y_filtered(1:n/2));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Spectrum of Filtered Signal');
grid on;

% Resample the audio signal to a lower sampling rate
newFs = 16000;  % Target sampling rate (Hz)
resampledSignal = resample(filteredSignal, newFs, fs);

% Display new sampling rate
disp(['Resampled Sampling Rate: ', num2str(newFs), ' Hz']);

% Play the original and processed audio
disp('Playing original audio...');
sound(audioSignal, fs);
pause(length(audioSignal)/fs + 1);

disp('Playing filtered and resampled audio...');
sound(resampledSignal, newFs);

% Save the processed audio
audiowrite('processed_speech.wav', resampledSignal, newFs);

disp('Processed audio saved as processed_speech.wav');
```