Problem 1: Pre-processing for Speech Recognition

Speech Recognition is ubiquitous: just ask Google Assistant :-) To improve feature extraction and classification accuracy by their ML algorithms, pretty much all speech recognition systems must remove unwanted signals from the user's speech using DSP algorithms. These *pre-processing* algorithms help the app function with background noise from traffic, appliances, etc.

The objective of this problem is to design and implement a simple denoising algorithm in Matlab, using the concepts of STFT and DFT from signals of the form $y(n) = x(n) + w(n)$ where $x(n)$ is the speech signal, and $w(n)$ represents different models for undesirable signals, that we will just call "noise".

1. **Part 1 (Suppressing noise of a known type)** You are given a noisy recording of speech sampled at 8 kHz that is a speech recording where $w(n)$ consists of bursts of tones. You are required to design an algorithm that will locate the tone bursts and attenuate them.
2. **Part 2 (Suppressing noise from unknown bands)** In this second part the speech signal, $x(n)$, is additively corrupted with an undesirable noise signal which could lie in spectral bands of 200Hz starting at 0Hz ("DC"). You are required to identify the noisy band(s) in the speech signal using a suitable method and suppress it so that the overall perceptual quality of speech improves.
3. In each case, your code will need to play the corrupted and the cleaned up speech signals.

Design criteria:

- Your approach must avoid altering the speech content as much as possible.
- The speech files `speech_with_beeps.txt` and `speech_noisy.txt` are only *examples* to help with the design. These should *not* be construed as the only test vectors. Both the tone suppression and denoising should work with a wide variety of typical usecases and together, with some variation in the tone type/amplitude as well as the type of noise.
- It would be ideal if you made no assumptions about the speakers: their number, gender, and the language and content of their speech. If you do, such assumptions should be clearly stated in your slides.
- To produce each output sample for the tone suppression algorithm (**Part 1**) you should require no more than 20 ms of samples (i.e., 160 samples at 8 kHz). In other words you are allowed to look no more than 20 ms into the future to suppress the noise in the current sample. This buffering mimics real time operation as it puts a limit on the time-lag("latency") between the signal input and signal output.
- It's okay to process denoising in **Part 2** offline and replay the speech.
- Note that the design should be able to suppress a variety of undesirable signals from typical environments – i.e., not restricted to the specific signal in `speech_noisy.txt`. Also, it should not assume only a specific number and gender of the speaker. These may be verified during the demo.