Voluntary homework on topic presented in Lecture 8:

(Sample applications of digital signal processing)

Sample-based melody synthesis (1 point, exceptional solution 1.5 points)

- Synthesize some well-known melody using tone samples from the file *Tones.mat* or optionally other samples freely available on the web (or recorded by yourself).
    - Concatenate the tone samples to form the melody (about 30 tones). Pay attention to the correct duration of the tones. The durations are not arbitrary, usually these are fractions of some bacis length. For example: if the basic tone has duration 0.5s (8000 samples at sampling frequency Fs=16000Hz), the shorter tone last for 0.25s (4000 samples). In musical theory this is denoted as ‘a whole note’ / ‘a half note’ (and ‘quarter note’ etc.). The duration of the whole note determines the tempo of the song; shorter note reslts into the quicker the tempo.
    - Prevent discontinuities at the tone boundaries using suitable windowing. The recorded tones are not continuous when appended. Discontinuities in the aimplitude appear which manifest as ‚clicks‘ in the sound. The simplest way to prevent this phenomenon is to weight the tones by a suitable window, such that the tones begin and end close to zero.
    - Filter the melody by some room impulse response freely available on the web (RIR, see Lecture 1). Real-world RIRs are preferable in this scenario compared to their artificialy generated models (due to realism). This creates a sound effect that the sound originates from the environment, where the RIR was recorded. Mind the sampling frequency of the RIR, it needs to be the same as the sampling frequency of the melody.
        - Optionally, you may even create a stereo sound by filtering the melody by two-channel response (one RIR for each channel), this is the basic creation of spatialized sound. The utilized RIRs need to be stereo RIRs (i.e. recorded simulaneously in the same environment using the same hardware setup/microphone array geometry).

- Evaluation criteria
    - Originality
    - Functionality/runable + correctness of solution
    - Comments in code, clean code
    - Effective code (fast, robust w.r.t. input variations)