**Report**

**Group composition:** Riccardo Pomarico, Alfredo Brusca.

Q1

The short-time Fourier transform is the same as the Fourier transform but it is evaluated on a short window length. By plotting it in the time domain we have a representation of the evolution of the frequency spectrum over time.
The STFT chromagram is defined as the energy associated to every note. It is calculated by adding up the energy for the associated pitch for every octave.
The CQT is related to the Fourier transform but, unlike the Fourier transform, it uses a logarithmically spaced frequency axis. The main difference with the STFT chromagram is that the CQT chromagram is more effective if used for musical representation.

Q2

We compute the prominence as the summation of the chromagram along the time axis for each chroma index. Then we added the values of the prominence for each chroma index to the chroma_vals list. In the end we define key_vals, which is a dictionary that associates chroma_vals elements to each key.
The grouping operation allows to calculate the energy of a note during the whole musical piece, since we don't need the energy in a specific instant to calculate the key of the song.

Q3

Both in the case in which we consider binary profiles and in the case of perceptual profiles the algorithm detects a D major key, which doesn't match with the ground-truth that is G major. Between D major and G major there is only one note off.
Using the perceptual model we usually obtained a more accurate result, with this musical piece the problem may be caused by noise or untuned percussive instruments.

Q4

We followed the same steps as in the previous questions, and the most likely key is still D major. But adding to the algorithm the ability to calculate the second best key, we now obtained the right result which is G major, the same as the ground truth.

Q5

Applying both binary and perceptual profiles to both the two types of chromagram for different tracks gives us the opportunity to observe that the CQT chromagram returns a correct output both using binary and perceptual profiles. The only exception is in Vivaldi's Allegro in the case of using a binary profile with CQT chromagram, in fact, the ground truth key is F minor but the detected key is G# major. But G# major is the relative major of F minor, which in this case turns out to be the alt key.
Applying the binary profile to the STFT chromagram gives us the wrong result in most cases: it works only in Blue Christmas and as an alternative key in Allegro.
Applying the perceptual profile to the STFT chromagram allows us to receive more accurate results: the outputs of the various tests match the ground truth, the only exceptions are in Allegro, where the correct key is the alternative one, and in Blue Christmas where the output is wrong (the alternative key is one note away from the true key).
So applying the perceptual profile to CQT chromagram is the best option because it is the one that always gives us the correct output.
The chromagram plot shows that the CQT is dense and less noisy, thus it always works better for the audio recording provided as they are all analog recordings with real instruments whose pitch fluctuates.

A strategy to improve the algorithm's performance is to normalize, smooth, and downsample the chromagram. In this way, we are able to detect the right key by applying the binary profile to the STFT chromagram even with Sweet home Alabama.
An example of what the function compute_chromagram would be is:

```
def compute_chromagram(x, Fs, chroma_type='stft',
print_chromagram=True):
    """Compute the chromagram according to the chroma_type

    Args:
        x: input audio file .wav
        Fs: sampling frequency
        chroma_type: type of the chromagram (default='stft')
        print_chromagram: if True, print chromagram (default=True)
    Return:
        chromagram: computed chromagram
    """

    def compute_SM_dot(X,Y):
        S = np.dot(np.transpose(Y),X)
        return S

    x_duration = (x.shape[0])/Fs

    N, H = 4096, 1024

    # Compute the chromagram

    if chroma_type == 'stft':
        chromagram = librosa.feature.chroma_stft(y=x, sr=Fs, tuning=0,
norm=2, hop_length=H, n_fft=N)
        X, Fs_X =
ssm_utils.smooth_downsample_feature_sequence(chromagram, Fs/H,
filt_len=41, down_sampling=10)
        X = ssm_utils.normalize_feature_sequence(X, norm='2',
threshold=0.001)
        S = compute_SM_dot(X,X)

    # Plot the chromagram
    if print_chromagram:

        plt.figure(figsize=(6, 6))
        cmap = ssm_utils.compressed_gray_cmap(alpha=-10)
        im = plt.imshow(S, aspect='equal', origin='lower', cmap=cmap,
vmin=0, vmax=1)
        plt.tight_layout()

    return chromagram
```