```
void ece420ProcessFrame(sample_buf *dataBuf) {
    isWritingFft = false;

    // Keep in mind, we only have 20ms to process each buffer!
    struct timeval start;
    struct timeval end;
    gettimeofday(&start, NULL);

    // Data is encoded in signed PCM-16, little-endian, mono channel
    float bufferIn[FRAME_SIZE];
    for (int i = 0; i < FRAME_SIZE; i++) {
        int16_t val = ((uint16_t) dataBuf->buf_[2 * i]) | (((uint16_t) dataBuf->buf_[2 * i + 1]) << 8);
        bufferIn[i] = (float) val;
    }

    // Spectrogram is just a fancy word for short time fourier transform
    // 1. Apply hamming window to the entire FRAME_SIZE
    // 2. Zero padding to FFT_SIZE = FRAME_SIZE * ZP_FACTOR
    // 3. Apply fft with KISS_FFT engine
    // 4. Scale fftOut[] to between 0 and 1 with log() and linear scaling
    // NOTE: This code block is a suggestion to get you started. You will have to
    // add/change code outside this block to implement FFT buffer overlapping (extra credit part).
    // Keep all of your code changes within java/MainActivity and cpp/ece420_*
    // ******************** START YOUR CODE HERE ********************** //
    // thread-safe
    isWritingFft = true;


    float pi = 3.1415926535;
    float window[FRAME_SIZE];
    for (int i =0;i<FRAME_SIZE;i++){
        window [i] = (0.54-0.46*cos((2*pi*i)/(FRAME_SIZE-1)));
    }
    //initialize data array as zeros so that after data is loaded in, we have trailing zeros as padding
    float data[FRAME_SIZE*2]={0};
    for (int j=0;j<FRAME_SIZE;j++){
        data[j] = bufferIn[j]*window[j];
    }

    kiss_fft_cpx fin[FFT_SIZE];
    kiss_fft_cpx fout[FFT_SIZE];
```

```

    for (int k=0;k<FFT_SIZE;k++){
        fin[k].r = data[k];
    }
    kiss_fft_cfg cfg = kiss_fft_alloc(FFT_SIZE,0,NULL,NULL);
    kiss_fft(cfg,fin,fout);
    for (int l=0;l<FRAME_SIZE;l++){
        fftOut[l] = log(fout[l].i*fout[l].i+fout[l].r*fout[l].r)/20;
    }



//    // Currently set everything to 0 or 1 so the spectrogram will just be blue and red stripped
//    for (int i = 0; i < FRAME_SIZE; i++) {
//        fftOut[i] = (i/20)%2;
//    }

    // ******************** END YOUR CODE HERE ********************** //
    // Flip the flag so that the JNI thread will update the buffer
    isWritingFft = false;

    gettimeofday(&end, NULL);
    LOGD("Time delay: %ld us",  ((end.tv_sec * 1000000 + end.tv_usec) - (start.tv_sec * 1000000 + start.tv_usec)));
}

```