DSP Lab Demo 12 Exercise 1

Major Code Changes

1. Implemented Block Processing

```
BLOCKLEN = 1024
for i in range(0, num_blocks):
    input_bytes = wf.readframes(BLOCKLEN)
    input_tuple = struct.unpack('h' * BLOCKLEN, input_bytes)

    # process each sample in the block
    for n in range(0, BLOCKLEN):
        ...

    output_bytes = struct.pack('h' * BLOCKLEN, *output_block)
    stream.write(output_bytes)
```

2. Used Canonical Form Filter Structure

```
M = max(len(a), len(b)) - 1
w = [0.0] * (M + 1)  # w[1..M] are delay states

w0 = x - sum(a[k]*w[k] for k=1..M)
y = sum(b[k]*w[k] for k=0..M)  # where w[0] := w0

for k in range(M, 0, -1):
    w[k] = w[k-1] if k-1 >= 1 else w0
```

3. Maintained Filter State Across Blocks

```
w = [0.0] * (M + 1)
```

Outcome

The block-based implementation produces the same filtered audio output as the single-sample version while being more efficient. Block duration of 1024 samples at 44.1 kHz provides a good balance between latency and efficiency. The canonical form requires only M delay states instead of 2M, which reduces memory usage by half.