1: The First Problem

$$\begin{aligned} x(t) &: \quad non-periodic. \\ CTFT &: \quad x(jw) = \int_{-\infty}^{\infty} x(t)e^{-jwt}dt \\ &: \quad x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} x(jw)e^{jwt}dw \end{aligned}$$

2: The Second Problem

3: The Third Problem

Unit Impulse is a sequence which has only one nonzero value, that occurs at n = 0.

$$\delta[n] = \begin{cases} 1, & \text{if } n = 0, \\ 0, & \text{if } n \neq 0. \end{cases}$$

Unit Impulse Response is the output sequence, when the input to the FIR filter is a unit impulse sequence.

$$h[n] = \textstyle\sum_{k=0}^{M} b_k \delta[n-k] = \begin{cases} 1, & \text{if } n = 0, \\ 0, & \text{if } n \neq 0. \end{cases}$$

4: The Fourth Problem

There are two general LTI system proporties: **time invariant** and **linearity**.

Time Invariant:

$$\begin{aligned} x[n] &\longmapsto y[n] \\ x[n-n_0] &\longmapsto y[n-n_0] \end{aligned}$$

Linearity:

$$\begin{aligned} x_1[n] &\longmapsto y_1[n] \\ x_2[n] &\longmapsto y_2[n] \\ x[n] = \alpha x_1[n] + \beta x_2[n] &\longmapsto y[n] = \alpha y_1[n] + \beta y_2[n] \end{aligned}$$

Derivation of Comvolutional Sum:

We can represent any signal as sum of impulses:

$$\begin{aligned} x[n] &\longmapsto \boxed{\text{S}} \longmapsto y[n] \\ x[n] &= \sum_{k=-\infty}^{\infty} x[n]\delta[n-k] \end{aligned}$$

The output will be:

$$\begin{aligned} y[n] &= S[x[n]] \\ y[n] &= S[\sum_{k=-\infty}^{\infty} x[n]\delta[n-k]] \\ y[n] &= \sum_{k=-\infty}^{\infty} x[n]S[\delta[n-k]] \qquad \text{(linearity)} \end{aligned}$$

Now, we need to know $S[\delta[n-k]]$, we can assume that this equation can be written safely:

$$\begin{aligned} h[n] &= S[\delta[n]] \\ h[n-k] &= S[\delta[n-k]] \qquad \text{(time invariant)} \end{aligned}$$

All these equations above give us the convolutional sum formula:

$$y[n] = \sum_{k=-\infty}^{\infty} x[n]h[n-k]$$

5: The Fifth Problem

(a)

Not casual: The function is dependent on future
Stable: x and y can be bounded.

(b)

Casual: The function is independent on future
Not stable: x and y cannot be bounded.

(c)

Casual: The function is independent on future
Stable: The function cannot exceed 1.

(d)

Casual: The function is independent on future
Stable: The function cannot exceed 1.

6: The Sixth Problem

I designed a function named **my_conv** which takes two parameters, signal array ($x[n]$) and impulse response ($h[n]$), and returns the convolution.

```
Emre:Homework2 KEO$ cat my_conv.py

def my_conv(x, h):
    m = [[0] * (len(x) + len(h) - 1) for _ in range(len(h))]
    y = [0] * (len(x) + len(h) - 1)

    for i in range(len(h)):
        for j in range(len(x)):
            m[i][i+j] = h[i] * x[j]

    for i in range(len(x) + len(h) - 1):
        for j in range(len(h)):
            y[i] += m[j][i]

    return y


if __name__ == '__main__':
    X = [2, 4, 6, 4, 2]
    H = [3, -1, 2, 1]

    print(my_conv(X, H))

Emre:Homework2 KEO$ python my_conv.py
[6, 10, 18, 16, 18, 12, 8, 2]
```

Figure 1: The output of my_conv function for given parameters in question

7: The Seventh Problem

$$\begin{aligned}
y[n] &= x[n] * h[n] \\
y[n] &= \begin{bmatrix} 2 & 4 & 6 & 4 & 2 \end{bmatrix} * \begin{bmatrix} 3 & -1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & -1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & -1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & -1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 & -1 & 2 & 1 \end{bmatrix} \\
y[n] &= \begin{bmatrix} 6 & 10 & 18 & 16 & 18 & 12 & 8 & 2 \end{bmatrix}
\end{aligned}$$

8: The Eighth problem

The convolve2d method in the scipy library should take at least two parameter, array likes inputs and convolve this two 2-dimensional arrays.

When the program executed, the resulting image, called "result.png" will be created in the very same folder.

```
[Emre:Homework2 KEO$ cat convolution.py

 import numpy
 import scipy.signal
 import matplotlib.image as mpimg
 from skimage.color import rgb2gray

 K = numpy.array([[1 / 9.] * 3 for i in range(3)])
 img = mpimg.imread('noisyCameraman.png')

 convolved = scipy.signal.convolve2d(rgb2gray(img), K, mode='same', boundary='fill', fillvalue=0)
 scipy.misc.imsave('result.png', convolved)

[Emre:Homework2 KEO$ python3 convolution.py
```

Figure 2: The usage of written python program

Figure 3: The resulting image

As we can see the resulting image in Figure 3, convolved image is smoother.