Using MATLAB (m-file):

In this home work, we want to generate a PCM modulator using m-file, to do this, follow the following:

a) Generate a sin wave \( x = \sin(t) \) and plot it with the rang \( t=0:0.01:10 \);

```matlab
clc
close all
clear all
%% analog signal
% t=0:0.01:10;
x=sin(t);
subplot(211)
plot(t,x)
grid
```

b) Perform a uniform quantization with 4 and 8 levels to the signal \( x \), and plot the output of each one in the same figure with \( x \). you can use the function \( uquant \) which included with this home work.

*For 4 level

```matlab
%% quantization
Quantized_signal=uquant(x,4);
subplot(212)
plot(t,Quantized_signal)
grid
```

* For 8 level

```matlab
%% quantization
Quantized_signal=uquant(x,8);
subplot(212)
plot(t,Quantized_signal)
grid
```
c) Perform a binary coding to the quantized signal.

```matlab
%% Encoding
bit=de2bi(Quantized_signal,'left-msb');
bit=bit';
bit=bit(:);
bit=bit'
```


d) Plot the first 10 bit from your output bit stream.

```matlab
figure(2)
stem(bit)
axis([1 10 0 1])
grind
```

*For 4 level

* For 8 level

After that, and from MATLAB output, answer the following:

e) How many samples are there?

t=0:0.01:10;

\[
\text{# of samples} = \frac{10}{0.01} + 1 = 1001
\]

f) How many bits in your bit stream can you transmit with 4 and 8 quantization levels?

**In 4 levels:**

\[
\text{# of bits} = 2 \times \text{# of samples} = 2 \times 1001 = 2002
\]

**In 8 levels:**

\[
\text{# of bits} = 3 \times \text{# of samples} = 3 \times 1001 = 3003
\]
g) What can you say about your results in part (f)?
- Increasing number of bits will increase the accuracy of the encoded signal.
- Increasing number of bits will increase the bandwidth of the transmission.

h) How many symbols can you transmit with 4 and 8 quantization levels?

# of symbols depend on the modulation used in the transmission. Here there is no modulation technique required, so the system is binary transmission system.

So, in 4 & 8 levels, # of symbols in equal to # of bits.

**In 4 levels:**

# of symbols = 2002

**In 8 levels:**

# of symbols = 3003

i) Try to explain the function (uquant). After that and for 4 quantization levels, find the "uquant" output when the input samples are (0.7, 0.5, -0.4, -0.7)

```matlab
function y=uquant(x,n)
%defining the function in MATLAB
% This function perform a uniform quantization and required two parameters,
% the first one is the sample and the second is # of levels that we need
a=max(max(x))-min(min(x));
% this is a variable that compute the difference between the maximum value
% of the signal and the minimum one.
delta= a/(n-1);
% the value of the quantization step between levels.
z=(x-min(min(x)))/delta;
y=round(z);
% Round the sampled signal to the nearest level.
```

- Using this function above to find the output when the input sample is:

```
function y=uquant(x,n)
%defining the function in MATLAB
% This function perform a uniform quantization and required two parameters,
% the first one is the sample and the second is # of levels that we need
a=max(max(x))-min(min(x));
% this is a variable that compute the difference between the maximum value
% of the signal and the minimum one.
delta= a/(n-1);
% the value of the quantization step between levels.
z=(x-min(min(x)))/delta;
y=round(z);
% Round the sampled signal to the nearest level.
```

```
n = 4
max(max(x)) = 1
min(min(x)) = -1
a = 1—1= 2
delta = 2/3 = 0.667
z = (x — (-1))/0.667 = \( \frac{x + 1}{0.667} \)  ........... (1)
y = round(\( \frac{x + 1}{0.667} \))  ........... (2)
```
When $x = (0.7, 0.5, -0.4, -0.7)$, use equation 1 and 2, then

<table>
<thead>
<tr>
<th>Input (x)</th>
<th>Output (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>3</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>-0.4</td>
<td>1</td>
</tr>
<tr>
<td>-0.7</td>
<td>0</td>
</tr>
</tbody>
</table>

j) Give me a brief description about how you simulate the encoding stage using MATLAB.

```matlab
%% Encoding
bit=de2bi(Quantized_signal,'left-msb');
bit=bit';
bit=bit(:);
bit=bit'
```

- Assume that the quantized signal is $[1 2 3 0]$
- Quantized_signal = $[1 2 3 0]$
- bit =
  
  0 1  
  1 0  
  1 1  
  0 0  
- bit =
  
  0 1 1 0  
  1 0 1 0  
- bit =
  
  0  
  1  
  1  
  0  
  1  
  1  
  0  
- bit =
  
  0 1 1 0 1 1 0 0