In [20]:

```
import numpy as np
```

In [21]:

```
def Time_Decimation_FFT(x, N):
    if(N == 1):
        return x

    W = np.exp(-2j*np.pi/N)

    #Zero Padding
    signal = np.zeros(N)
    signal[:len(x)] = x

    #Time Decimation
    X_even = Time_Decimation_FFT(signal[0::2], N//2)
    X_odd = Time_Decimation_FFT(signal[1::2], N//2)

    X = np.zeros(N,dtype=complex)

    for k in range(N//2):
        X[k] = X_even[k] + (W**k)*X_odd[k]
        X[k + N//2] = X_even[k] - (W**k)*X_odd[k]

    return X
```

In [23]:

```
test = np.random.rand(16)  #Random array generator
npfft = abs(np.fft.fft(test, 32))  #numpy FFT
myfft = np.abs(Time_Decimation_FFT(test, 32))  #My FFT

print('numpy = {}'.format(npfft))
print('mine = {}'.format(myfft))
```

```
numpy = [9.45066285 6.09972375 1.45494557 1.35615052 1.29535051 1.67799643
 0.68457289 1.41903325 0.85234416 1.36134014 1.00137807 1.6476452
 0.73100514 2.01530629 1.16392923 0.19154968 0.06215652 0.19154968
 1.16392923 2.01530629 0.73100514 1.6476452  1.00137807 1.36134014
 0.85234416 1.41903325 0.68457289 1.67799643 1.29535051 1.35615052
 1.45494557 6.09972375]
mine = [9.45066285 6.09972375 1.45494557 1.35615052 1.29535051 1.67799643
 0.68457289 1.41903325 0.85234416 1.36134014 1.00137807 1.6476452
 0.73100514 2.01530629 1.16392923 0.19154968 0.06215652 0.19154968
 1.16392923 2.01530629 0.73100514 1.6476452  1.00137807 1.36134014
 0.85234416 1.41903325 0.68457289 1.67799643 1.29535051 1.35615052
 1.45494557 6.09972375]
```