Name	**Pranay Singhvi**
UID no.	**2021300126**

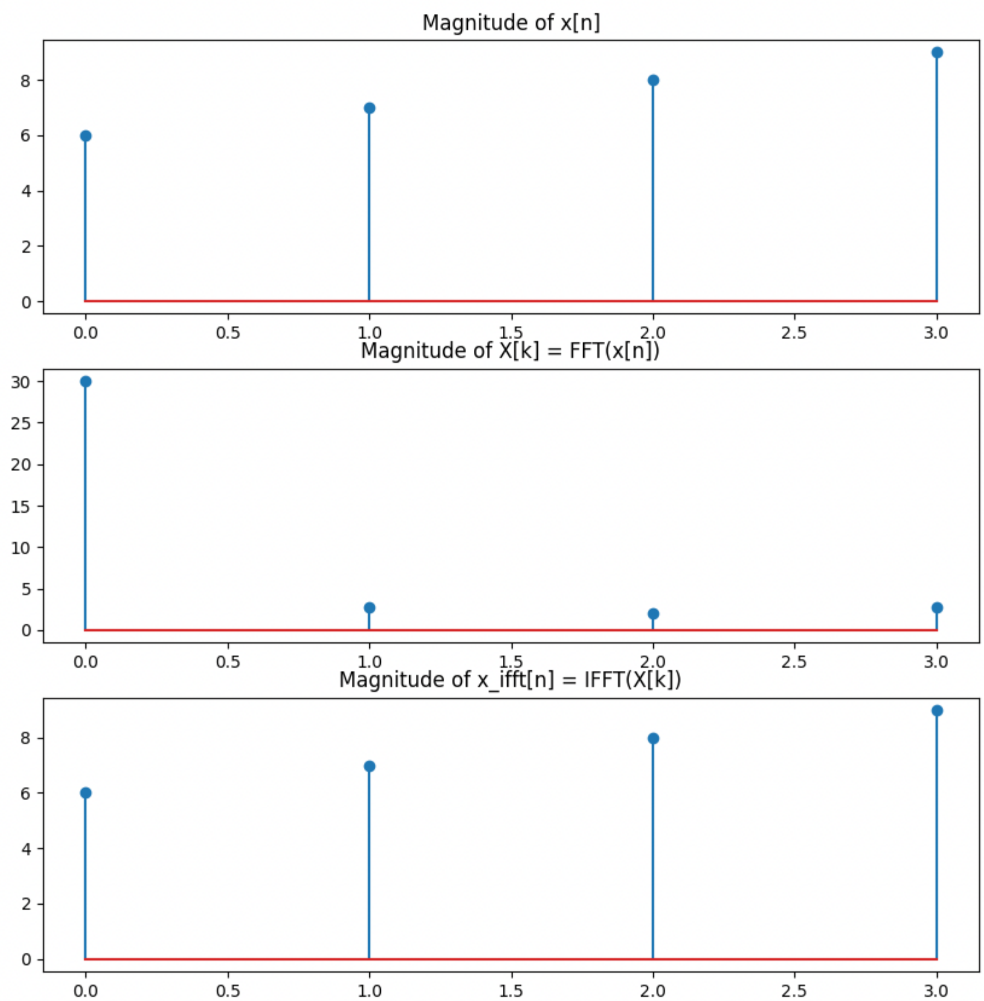
Experiment 4	
AIM:	The aim of this experiment is to implement computationally Fast Algorithms.
OBJECTIVE:	1.Develop a program to perform FFT of N point Signal. 2.Calculate FFT of a given DT signal and verify the results using mathematical formula. 3.Computational efficiency of FFT.
PROBLEM DEFINITION:	(1) Take any four-point sequence x[n]. Find FFT of x[n] and IFFT of {X[k]}. (2) Calculate Real and Complex Additions & Multiplications involved to find X[k].
INPUT SPECIFICATIONS	**1.** Length of first Signal N 2. DT Signal values

EXPERIMENTATION AND RESULT ANALYSIS

CASE 1: To find DFT of 4 point sequence

Input x[n] = { 6,7,8,10 } Length L= 4

Output : x[K] =



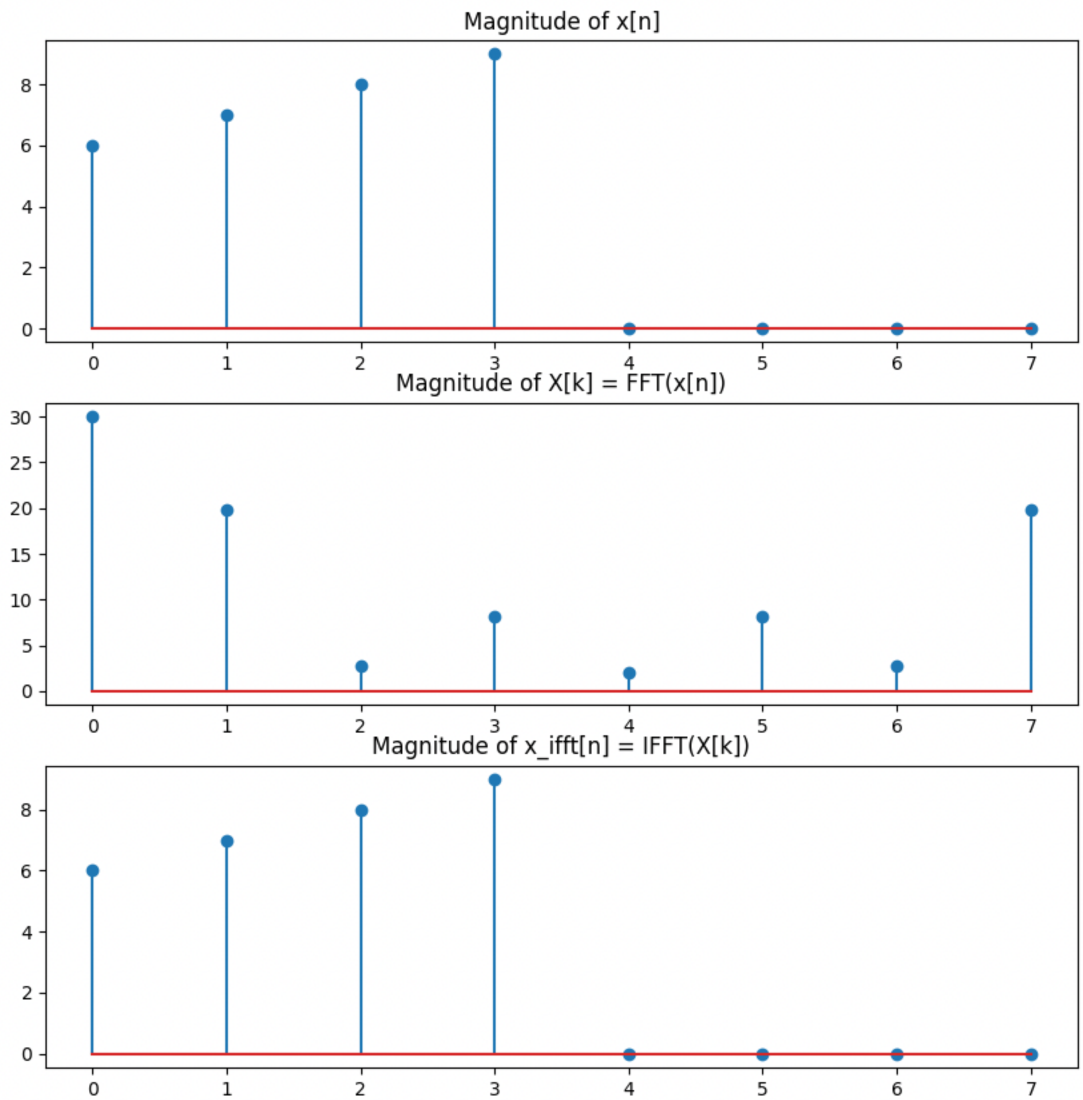
```
Enter the length of the signal (N): 4
Enter the value of x[0]: 6
Enter the value of x[1]: 7
Enter the value of x[2]: 8
Enter the value of x[3]: 9
Magnitude of FFT output: [30.          2.82842712  2.          2.82842712]

Operation Counts for FFT:
Complex Additions: 8
Complex Multiplications: 4

Time Comparison:
DFT time: 0.0008487701416015625 seconds
FFT time: 0.0014498233795166016 seconds
```

CASE 2 To find DFT of zero padded signal

Input x[n] = { 6,7,8,10,0,0,0,0 } Length L= 8



```
Enter the length of the signal (N): 8
Enter the value of x[0]: 6
Enter the value of x[1]: 7
Enter the value of x[2]: 8
Enter the value of x[3]: 9
Enter the value of x[4]: 0
Enter the value of x[5]: 0
Enter the value of x[6]: 0
Enter the value of x[7]: 0
Magnitude of FFT output: [30.         19.85066178  2.82842712  8.12103606  2.          8.12103606
  2.82842712 19.85066178]

Operation Counts for FFT:
Complex Additions: 24
Complex Multiplications: 12

Time Comparison:
DFT time: 0.0009331703186035156 seconds
FFT time: 0.0019440650939941406 seconds
```

CONCLUSION:

1. Computational Efficiency in DFT:
a) Total Real Multiplications = 4N^^2
b) Total Real Additions = 4N^^2-2N
2. Computational Efficiency in FFT:
a) Total Real Multiplications = 2N*Log2N
b) Total Real Additions = 3N*Log2N
3. FFT produces fast results due to:
a) Less Computations
b) Parallel implementations