Exercise N7. Verify by direct hand calculation that $F_{N}^{-1} = N F_{N}$ in the case that $N = 3$.

Solution. (8 pts) We use the fact that $\omega_N = \omega^{-1}$ and calculate

$$F_3 N F_3 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega_N & \omega_N^{-2} \\ 1 & \omega_N^2 & \omega_N^{-4} \end{bmatrix} \cdot \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega_N & \omega_N^2 \\ 1 & \omega_N^2 & \omega_N^4 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 + 1 + 1 & 1 + \omega_N + \omega_N^2 & 1 + \omega_N^2 + \omega_N^4 \\ 1 + \omega_N^{-1} + \omega_N^{-2} & 1 + 1 + 1 & 1 + \omega_N + \omega_N^2 \\ 1 + \omega_N^2 + \omega_N^4 & 1 + \omega_N + \omega_N^2 & 1 + 1 + 1 \end{bmatrix}.$$

Here we use the fact that if $\omega^3 = 1$, then

$$(1 + \omega + \omega^2)(1 - \omega) = 1 - \omega^3 = 0,$$

which implies that $1 + \omega + \omega^2 = 0$ and makes all the off diagonal terms zero.

Exercise N8. Make out a table with two columns. On the left write the binary format of integers 0 through 3, then in the second column reflect the binary form and convert the result to decimal numbers. Repeat this with integers 0 through 7. What does this tell you about the ordering for the indices of the FFT?

Solution. (8 pts) We get

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>

From what we see and compare to the ordering in the discussion of the FFT for $N = 8$, that reversing the binary digits results in the correct ordering for FFT induction.

Exercise N9. Compare the flop count for computing the product of a polynomial of degree 1000 and a polynomial of degree 2 by direct calculation and by the FFT method. Who wins?

Solution. (8 pts) For the FFT, we have to pad to 1024 elements and get a cost of $1024 \cdot 10$ additions and about half that number of multiplications for a total cost of about $1024 \cdot 15 = 15360$ flops. If we do it directly, the cost is only about $1000 \cdot 6 = 6000$ total flops, so direct multiplication wins.

Exercise N10. Write a Matlab routine that implements FFTsort.

Solution. (8 pts) We have the following:
function retval = FFTsort(p)
% usage: ndx = FFTsort(p)
% description: given positive integer p as input,
% returns an index vector of length N = 2^p
% suitable for implementing the FFT, with
% indexing starting at 1.
N = 2^p;
retval = (1:N);
stride = N;
for k = p:-1:1
    for j = 1:stride:N-1
        m = j+stride-1;
        retval(j:m) = [retval(j:2:m),retval(j+1:2:m)];
    end
    stride = stride/2;
end

Exercise N11. Use your FFTsort program and FFTTransform to confirm in Matlab the claim that \( y_k = N \mathcal{F}_N(\tilde{Y}_n) \) with a few calculations involving data vectors of length 4, 8, 16. Solution. (8 pts) Check it. It works.