```
function [ Ff ] = FFT( f )

    %%WARNING: works ONLY if length(f) = 2^n!!!

    N = length(f);
    Ff = zeros(2*N,1); %it will shrink at the end, extra zeroes are workspace

    fFirstHalf = f(1:2:length(f)-1);
    fSecondHalf = f(2:2:length(f));

    %odd entries in the first half of the vector, separeted by a zero, even
    %entries in the second half
    for k = 1:1:N
        if k <= N/2
            Ff(2*k - 1) = fFirstHalf(k);
        else
            Ff(2*k - 1) = fSecondHalf(k - N/2);
        end
    end

    %at step n subdivide the vector in subvectors with lengths 2^n.
    for k = 1:1:log2(2*N) - 1

        %to every subvector apply a dilation
        for h = 0:2^k:2 * N - 1
            Ff(h + 1:h + 2^k) = D(Ff(h + 1:h + 2^(k-1)));
        end

        %to every even subvector apply a modulation
        for h = 1:1:2*N/2^k
            if not(mod(h,2) == 0)
                Ff(h*2^k + 1:(h + 1)*2^k) = M(1,Ff(h*2^k + 1:(h + 1)*2^k));
            end
        end

        %after that, sum every couple of subvectors and keep just this sum
        for h = 0:2^(k + 1):2 * N - 1
            Ff(h + 1:h + 2^k) = Ff(h + 1:h + 2^k) + Ff(h + 2^k + 1:h + 2^(k + 1));
            Ff(h + 2^k + 1:h + 2^(k + 1)) = zeros(2^k,1);
        end

    end

    Ff = N*flipud(Ff(1:N)); %normalization term

    %modulation function
    function Mg = M( m,g )

        n = length(g);
        Mg = zeros(n,1);

        for j = 1:1:n
            Mg(j) = g(j)*exp(2i * pi * m * j / n);
        end

    end

    %dilation function
    function Dg = D( g )
```

```
        Dg = 0.5*[g;g];
    end
end
```