```
#include <complex.h>
#include <nfft/nfft3.h>

/* Computation of the positive frequency
 * part of the (unnormalised) Fourier
 * transform of a times-series (t, y).
 *
 * Input:
 *  t  the times reduced to [1/2, 1/2)
 *  y  the measurements (NULL, for
 *     computing the FT of the window)
 *  n  the number of measurements
 *  m  the number of positive frequencies
 * Output:
 *  d  the Fourier coefficients
 *     (preallocated array for (m+1)
 *     elements)
 */
void nnft(const double* t, const double* y,
          int n, int m, double complex* d)
{
  // Creates NFFT plan for 2*m Fourier
  // coefficients (positive and negative
  // frequencies) and n data samples.
  nfft_plan p;
  nfft_init_1d(&p, 2 * m, n);

  if (y != NULL) // data spectrum
  {
    for (int i = 0; i < n; i++)
    {
      p.x[i] = t[i];
      p.f[i] = y[i];
    }
  }
  else           // window spectrum
  {
    for (int i = 0; i < n; i++)
    {
      p.x[i] = t[i];
      p.f[i] = 1.0;
    }
  }

  // Possibly optimises.
  if (p.nfft_flags & PRE_ONE_PSI)
    nfft_precompute_one_psi(&p);

  // Computes the adjoint transform.
  nfft_adjoint(&p);

  // Outputs the positive frequency
  // Fourier coefficients.
  for (int i = 0; i < m; i++)
    d[i] = p.f_hat[i];
  d[m] = conj(p.f_hat[0]);

  nfft_finalize(&p);
}
```