```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
int ibitr(int, int);
void fft(double [], double [], int ,int);
void fft(double xr[], double xi[], int nu,int ie)
{
static int n,n1,n2,nu1,p,k1n2,i,k,l;
static double arg,c,s,tr,ti;
double *stab, *ctab;
#define TWOPI 6.2831853  /* 2 PI */
#define PITWO 1.5707963   /* PI/2 */
/* allocate storage for sine and cosine tables */
n=1<<nu;
stab = (double *) calloc(n, sizeof(double));
ctab = (double *) calloc(n, sizeof(double));
if ( (stab == NULL) || (ctab == NULL) )
     {
     printf(" Can't allocate fft.c data storage - exit\n");
     exit(1);
     }
n2=n/2;
nu1=nu-1;
for (i=0; i<n; ++i)
     {
     arg=TWOPI*i/n;
     stab[i]=sin(arg);
     ctab[i]=sin(arg+PITWO);
     }
k=0;
for (l=0; l<nu; ++l)
     {
     while (k<n)
     {
     for(i=0; i<n2; ++i)
          {
          n1=1<<nu1;
          p=ibitr(k/n1,nu);
          s=stab[p];
          c=ctab[p];
          if(ie>0) s=-s;
          k1n2=k+n2;
          tr = xr[k1n2]*c + xi[k1n2]*s;
```

```
          ti = xi[k1n2]*c - xr[k1n2]*s;
          xr[k1n2] = xr[k] - tr;
          xi[k1n2] = xi[k] - ti;
          xr[k] = xr[k] + tr;
          xi[k] = xi[k] + ti;
          k=k+1;
          }    /* end i loop  */
     k=k+n2;
     }         /* end while  */
     k=0;
     nu1=nu1-1;
     n2=n2/2;
     }         /* end l loop  */
for (k=0; k<n; ++k)
     {
     i=ibitr(k,nu);
     if(i>k)
          {
          tr=xr[k];
          ti=xi[k];
          xr[k]=xr[i];
          xi[k]=xi[i];
          xr[i]=tr;
          xi[i]=ti;
          }
     }
if (ie>0)
     {
     for(i=0; i<n; ++i)
          {
          xr[i]=xr[i]/n;
          xi[i]=xi[i]/n;
          }
     }
free(stab);
free(ctab);
return;
}

int ibitr(int j, int nu)
{
int rm,lm,i,bitr;
rm=1;
```

```
lm=1<<(nu-1);
bitr=0;
for (i=0; i<nu; ++i)
     {
     if ((j&rm)!=0) bitr=bitr|lm;
     rm=rm<<1;
     lm=lm>>1;
     }
return (bitr);
}
```