```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


#define NSIZE 160

int spin[NSIZE][NSIZE];

void Metropolis(double, int);
double Energy(void);
double ExactEnergy(double);
double Magnetization(void);
double realrand(void);

int
main()
{
   int i, j, k, nthermal, naverage, nsweep;
   double beta, TotalEnergy, TotalMagnetization;
   double betaini, betafin, betastep;

   nthermal=50;
   naverage=200;
   nsweep=10;
   betaini=0.435;
   betafin=0.45;
   betastep=0.002;

   // Scelgo la direzione iniziale degli spin a caso
   for(i=0; i<NSIZE; i++) {
     for(j=0; j< NSIZE ; j++) {
        if(realrand()>0.5) {
            spin[i][j] = 1;
         }
         else {
           spin[i][j] = -1;
         }
      }
   }

   printf("\nReticolo degli spin %d x %d\n",NSIZE, NSIZE);
   printf("Passaggi di termalizzazione %d\n",nthermal);
   printf("Media su %d stati\n",naverage);
   printf("calcolata su stati dopo %d passate\n",nsweep);

   fflush(stdout);

   // termalizzazione preliminare
   Metropolis(betaini,5*nthermal);
   for(beta=betaini; beta<=betafin; beta += betastep) {
       // termalizzazione
      Metropolis(beta,nthermal);

      // azzero i valori medi
      TotalEnergy=0;
      TotalMagnetization=0.;
      for(k=1; k<=naverage; k++) {
         Metropolis(beta,nsweep);
         // Calcolo dei valori indipendenti dell'energia
         // e della magnetizzazione
         // Quindi li sommo.
         TotalEnergy += Energy();
         TotalMagnetization += Magnetization();
      }
      // prendo i valori medi
      TotalEnergy /= ((double) naverage);
      TotalMagnetization /= ((double) naverage);

      // stampo i risultati
      printf("\n beta: %7.4lf  E: %7.4lf E esatta: %7.4lf Magnetizzazione %7.4lf",
          beta, TotalEnergy, ExactEnergy(beta), fabs(TotalMagnetization));
      // printf("%7.4lf %7.4lf\n",beta,fabs(TotalMagnetization));
   }
```

```
   printf("\n\n");
   return 0;
}

double
Energy()
{
   int isum, i, j;
   // int ip1, im1, jp1, jm1;

   // contributo di ogni legame all'energia. Ogni legame
   // viene contato una sola volta anziche' due,
   // per cui il risultato finale va moltiplicato per due
   isum=0.;
   for(i=0; i<NSIZE; i++) {
      for(j=0; j<NSIZE-1; j++) {
         isum += spin[i][j]*spin[i][j+1];
      }
      isum += spin[i][NSIZE-1]*spin[i][0];
   }
   for(j=0; j<NSIZE; j++) {
      for(i=0; i<NSIZE-1; i++) {
         isum += spin[i][j]*spin[i+1][j];
      }
      isum += spin[NSIZE-1][j]*spin[0][j];
   }
   isum=2*isum;


   // Energia senza campo magnetico
   return -( (double) isum)/( (double) NSIZE*NSIZE );
}

double
Magnetization()
{
   int isum, i, j;

   isum=0;
   for(i=0; i<NSIZE; i++){
      for(j=0; j< NSIZE; j++) {
         isum += spin[i][j];
      }
   }
   return ( (double) isum)/( (double) NSIZE*NSIZE );
}

void
Metropolis(double beta,int nsweep)
{
   // esegue n "passate" con l'algoritmo di Metropolis su tutto il reticolo

   int i, j, k, m, im1, ip1, jm1, jp1;
   int DeltaE;
   double f4,f2;

   f4=exp(-8*beta);    // k=  4
   f2=exp(-4*beta);    // k=  2

   // loop per NSIZE spin l volte
   for(m=1; m<= nsweep; m++) {
      for(i=0; i< NSIZE; i++) {
         ip1=(i+1)%NSIZE;
         im1=i-1;
         if(i==0) im1=NSIZE-1;

         for(j=0; j< NSIZE; j++) {
            jp1=(j+1)%NSIZE;
            jm1=j-1;
            if(j==0) jm1=NSIZE;
            k=spin[ip1][j]+spin[im1][j]+
                spin[i][j+1]+spin[i][j-1];
            DeltaE= 2*spin[i][j]*k;
            if(DeltaE <= 0) {
```

```
                spin[i][j] = -spin[i][j];
            }
            else {
               switch(abs(k)) {
                  case 4:
                     if(f4 > realrand()) spin[i][j] = -spin[i][j];
                     break;
                  case 2:
                     if(f2 > realrand()) spin[i][j] = -spin[i][j];
                     break;
               }
            }
         }
      }
   }
}

double realrand()
{
   static long int a=16807, m=2147483647, q=127773, r=2836, k;
   static int ncall=0;
   double fac=1./2147483647.;

   if(ncall==0) {
     ncall=1;
     k=198725;
   }

   k=a*(k%q) - r*(k/q);   // moltiplicazione modulo m
   if(k < 0) k += m;       // se il risultato e' negativo
                           // aggiungo m
   return fac*k;
}

double ExactEnergy(double beta)
{
   int j;
   double q1, q2, pihalf, pigrec, h, sum, x;

   q1=2*sinh(2*beta)/pow(cosh(2*beta),2);
   q2=2*pow((tanh(2*beta)),2)-1.;
   pihalf=2*atan(1.);
   pigrec=2*pihalf;
   h=pihalf/1000.;

   // questo e' un integrale che compare nella
   // formula analitica per l'energia
   sum=0.;
   for(j=0; j<=1000; j++) {
      x=j*h;
      sum += 1.0/sqrt(1.0-pow(q1*sin(x),2));
   }
   sum=sum-0.5*(1.+1./sqrt(1.-q1*q1));
   sum=sum*h;

   return -2./tanh(2*beta)*(1+q2/pihalf*sum);
}
```