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


#define NSIZE 100

int spin[NSIZE];

void Metropolis(double, int);
double Energy(void);
double ExactEnergy(double);
long int lcm1(void);
long int lcm2(void);
double realrand1(void);
double realrand2(void);
extern void r250init_1(void);
extern void r250init_2(void);
extern double r250_1(void);
extern double r250_2(void);

int
main() {

  int j, k;
  double beta, TotalEnergy, naverage=200;

  r250init_1();
  r250init_2();

  // Scelgo la direzione iniziale degli spin a caso
  for(j=0; j< NSIZE ; j++) {
    if(realrand1()>0.5) {
      spin[j] = 1;
    }
    else {
      spin[j] = -1;
    }
  }

  printf("\n lunghezza della catena di spin %d\n",NSIZE);
  fflush(stdout);

   printf("\n beta         E         E esatta\n");
   for(beta=0.1; beta<=3.0; beta += 0.1) {

    // termalizzazione per un certo valore di beta
    Metropolis(beta,500);


    // azzero i valori medi
    TotalEnergy=0;
    for(k=1; k<=naverage; k++) {
      Metropolis(beta,20);
      // Calcolo dei valori indipendenti dell'energia e li sommo
      TotalEnergy += Energy();
    }
    // prendo il valore medio
    TotalEnergy /= ((double) naverage);

    // stampo i risultati
    printf("%7.3lf   %7.3lf      %7.3lf\n",
          beta, TotalEnergy, ExactEnergy(beta));
  }
  printf("\n\n");
  return 0;
}

double
Energy()
{
  int isum, j;

  // Energia di interazione spin-spin
```

```
  isum=0;
  for(j=0; j< NSIZE-1; j++) {
    isum += spin[j]*spin[j+1];
  }
  isum += spin[NSIZE-1]*spin[0];
  // Energia senza campo magnetico
  return -( (double) isum)/( (double) NSIZE );
}

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

   long int j, k, i, im1, ip1;
   double f;

   f=exp(-4*beta);

   // loop per NSIZE spin l volte
   for(j=1; j<= n*NSIZE; j++) {
      // scelgo uno spin a caso tra 0 e NSIZE-1
      // i = realrand1() * (NSIZE-1);
      i = r250_1() * (NSIZE-1);
      // trovo i primi vicini di i
      im1=i-1;
      if(i==0) im1=NSIZE-1;
      ip1=i+1;
      if(i==NSIZE-1) ip1=0;
      // calcolo la variazione di energia che si avra' flippando lo spin
      k= 2*spin[i]*(spin[ip1]+spin[im1]);
      if(k <= 0) {
         spin[i] = -spin[i];
      }
      else {
         // if(f > realrand2()) {
         if(f > r250_2()) {
            spin[i] = -spin[i];
         }
      }
   }
}

double
ExactEnergy(double beta)
{
  // Calcola il rapporto tra energia e NSIZE dalle formule analitiche

  return -(exp(beta)-exp(-beta))/(exp(beta)+exp(-beta));
}

long int lcm1()
{
   static long int a=16807, m=2147483647, q=127773, r=2836, k;
   static int ncall=0;

   if(ncall==0) {
     ncall=1;
     k=812761;
   }

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

double realrand1()
{
   return ((double) lcm1())/2147483647.;
}

long int lcm2()
{
   static long int a=16807, m=2147483647, q=127773, r=2836, k;
```

```
   static int ncall=0;

   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 k;               // k e' l'attuale numero casuale
}

double realrand2()
{
   return ((double) lcm1())/2147483647.;
}
```