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

#define L 4
#define V L*L
#define MAX_ITER 100

#define DESTRA 0
#define ALTO 1
#define SINISTRA 2
#define BASSO 3

typedef unsigned long long RANDOM_TYPE;
typedef char SPIN;

#define MYRAND64  myrand64 = (6364136223846793005UL * myrand64)
#define MASSIMO_RAND64      0XFFFFFFFFFFFFFFFFUL
RANDOM_TYPE myrand64;
double inv_massimo_rand64 = 1.0L / (double)MASSIMO_RAND64;

unsigned long vicino[4][V];

struct s_cluster{
  SPIN spin;
  struct s_cluster *prossimo;
  struct s_cluster *precedente;
  struct s_cluster *trisavolo;
  struct s_cluster *trisnipote;
}cluster[V];

void my_init(void);
void init_vicini(void);
```

```c
void init_random_spin(void);
unsigned long site_update(long,long);
void my_end(void);

#define FAILURE -1
#define SUCCESS 1

/***************************************************************************
**/
int main(void)
{
  long i, j, done =0;
  unsigned long ho_cambiato;

  my_init();
  init_vicini();
  init_random_spin();
  for(j=0; j<MAX_ITER; j++){
    ho_cambiato = 0UL;
    for(i=0; i<V; i++){
      ho_cambiato += (unsigned long)site_update((long)DESTRA,i);
      ho_cambiato += (unsigned long)site_update((long)BASSO,i);
      ho_cambiato += (unsigned long)site_update((long)SINISTRA,i);
      ho_cambiato += (unsigned long)site_update((long)ALTO,i);
    }
    printf("# numero cambiamenti %ld\n",ho_cambiato);
    if(ho_cambiato==0){
      done = 1;
      break;
    }
  }
  my_end();
```

```
  if(done==0){
    printf("Non siamo riusciti a costruire il cluster\n");
    return FAILURE;
  } else {
    return SUCCESS;
  }
}

/**********************************************************************
**/
void my_end(void)
{

  long i,j;

  for(j=0; j<L; j++){printf("-----------");}printf("-\n");
  for(i=0; i<L; i++){
    for(j=0; j<L; j++){
      printf("| %2d (%d) ",
             cluster[j+i*L].spin,cluster[j+i*L].trisavolo-cluster);
    }
    printf("|\n");
    for(j=0; j<L; j++){printf("-----------");}printf("-\n");
  }
  printf("\n");

  printf(" ENDALL");
  for(i=0; i<L; i++){
    for(j=0; j<L; j++){
      printf(" | + %d - %d A %d Z %d",
             cluster[j+i*L].prossimo-cluster,
             cluster[j+i*L].precedente-cluster,
```

```
              cluster[j+i*L].trisavolo-cluster,
              cluster[j+i*L].trisnipote-cluster);
      }
      if(i<L-1)printf("|\n ENDALL");
    }
  printf("\n");

}

/***************************************************************************
**/
unsigned long site_update(long direzione, long sito)
{
  long i, j;
  unsigned long cambio=0UL;
  long sito_minore, sito_maggiore;
  struct s_cluster *testa_maggiore, *coda_maggiore;
  struct s_cluster *testa_minore, *coda_minore;
  struct s_cluster *puntatore;

  if((cluster[sito].spin==cluster[vicino[direzione][sito]].spin) &&
     (cluster[sito].trisavolo!=cluster[vicino[direzione][sito]].trisavolo)){

    /* qui si cambia */
    cambio = 1UL;

    /* copiero' dal sito con trisavolo piu' piccolo
       a quello con trisavolo piu' grande */
    if(cluster[sito].trisavolo<cluster[vicino[direzione][sito]].trisavolo){
      sito_minore = sito;
      sito_maggiore = vicino[direzione][sito];
    } else {
```

```
      sito_minore =  vicino[direzione][sito];
      sito_maggiore = sito;
    }
    /* siccome dovro' cambiare i valori conservo i dati che mi serviranno,
       come testa e coda dei due cluster che sto fondendo */
    testa_maggiore = cluster[sito_maggiore].trisavolo;
    coda_maggiore = cluster[sito_maggiore].trisnipote;
    testa_minore = cluster[sito_minore].trisavolo;
    coda_minore = cluster[sito_minore].trisnipote;
    /* l'antenato del cluster minore viene dato a tutti i siti
       del cluster maggiore */
    for(puntatore=testa_maggiore; ; puntatore=puntatore->prossimo){
      puntatore->trisavolo = testa_minore;
      if(puntatore==puntatore->prossimo)break;
    }
    /* il discendente del cluster maggiore viene dato a tutti i siti
       del cluster minore */
    for(puntatore=testa_minore; ; puntatore=puntatore->prossimo){
      puntatore->trisnipote = coda_maggiore;
      if(puntatore==puntatore->prossimo)break;
    }
    /*il prossimo della coda del cluster minore e' l'antenato
      del cluster maggiore (che stiamo assorbendo) */
    if(coda_minore->prossimo != coda_minore){
      printf("interruzione programma: errore interno 1\n");
      exit(-9);
    }
    coda_minore->prossimo = testa_maggiore;
    /*il precedente della coda del cluster maggiore e' il figlio
      del cluster minore (che sta prendendo il potere) */
    if(testa_maggiore->precedente != testa_maggiore){
      printf("interruzione programma: errore interno 2\n");
```

```
      exit(-9);
    }
    testa_maggiore->precedente = coda_minore;

  }
  return cambio;
}

/***************************************************************************
*/
void init_vicini(void)
{
  long x, y, i;

  for(x=0; x<L; x++){
    long xp, xm;
    xp = x+1; if(x==L-1)xp=0;
    xm = x-1; if(x==0)  xm=L-1;
    for(y=0; y<L; y++){
      long yp, ym;
      yp = y+1; if(y==L-1)yp=0;
      ym = y-1; if(y==0)  ym=L-1;
      vicino[DESTRA]  [x+L*y] = xp + L *  y;
      vicino[ALTO]    [x+L*y] =  x + L * yp;
      vicino[SINISTRA][x+L*y] = xm + L *  y;
      vicino[BASSO]   [x+L*y] =  x + L * ym;
    }
  }
  /* for(i=0; i<V; i++){
    printf("sito %d +x %d +y %d -x %d -y %d\n",
           i,vicino[0][i],vicino[1][i],vicino[2][i],vicino[3][i]);
           }*/
```

```
}

/**********************************************************************
**/
void init_random_spin(void)
{
  long i, j;

  for(i=0; i<V; i++){
    MYRAND64;
    if((double)myrand64*inv_massimo_rand64<.5){cluster[i].spin=-1;}
    else{cluster[i].spin=1;}
  }

  for(i=0; i<L; i++){
    for(j=0; j<L; j++){
      cluster[j+i*L].prossimo = cluster+(j+i*L);
      cluster[j+i*L].precedente = cluster+(j+i*L);
      cluster[j+i*L].trisavolo = cluster+(j+i*L);
      cluster[j+i*L].trisnipote = cluster+(j+i*L);
    }
  }

  for(j=0; j<L; j++){printf("-----");}printf("-\n");
  for(i=0; i<L; i++){
    for(j=0; j<L; j++){
      printf("| %2d ",cluster[j+i*L].spin);
    }
    printf("|\n");
    for(j=0; j<L; j++){printf("-----");}printf("-\n");
  }
  printf("\n");
```

```
  printf(" STAALL");
  for(i=0; i<L; i++){
    for(j=0; j<L; j++){
      printf(" | + %d - %d A %d Z %d",
             cluster[j+i*L].prossimo-cluster,
             cluster[j+i*L].precedente-cluster,
             cluster[j+i*L].trisavolo-cluster,
             cluster[j+i*L].trisnipote-cluster);
    }
    if(i<L-1)printf("|\n STAALL");
  }
  printf("\n");

}

/***************************************************************************
**/
void my_init(void)
{

  int random_seed;
  long i;

  printf("#Input random seed _ \n"); fflush(stdout);
  scanf("%d",&random_seed);
  printf("#random_seed = %d\n", random_seed); fflush(stdout);

  myrand64 = (RANDOM_TYPE)random_seed;
  for(i=0; i<1000; i++){
    MYRAND64;
    /*printf("%ld %llu %lg\n",
```

```
        i,myrand64,(double)myrand64*inv_massimo_rand64); */
   }
}
```