```
/*
 * Corso di Fondamenti di Informatica
 * Esercizio:
 * tema d'esame (2017-02-08)
 */

#include <stdio.h>
#define N 3

/* #define DEBUG */

/*
Si definisca un tipo â€œbooleanâ€ e un tipo atto a rappresentare una Matrice di N x N interi, con N predefinita tramite una
direttiva define.
Si scriva una funzione ricorsiva che, ricevendo in ingresso una Matrice e un eventuale altro parametro utile, restituisca un
valore boolean che vale true se la Matrice ricevuta in ingresso Ã¨ simmetrica, false in caso contrario.
*/

typedef enum {false, true} boolean;

typedef int Matrice[N][N];

boolean Simmetrica(Matrice Mat, int Dim);
/* riceve in ingresso una matrice di N x N elementi di cui considera la
 * sottomatrice di Dim x Dim elementi aventi indici che vanno da N-Dim a
 * N-1 e ne verifica la simmetria.
 * La funzione opera ricorsivamente: controlla se la prima riga e la prima
 * colonna della matrice ricevuta in ingresso sono uguali elemento per
 * elemento, e in caso lo siano demanda la verifica della parte rimanente
 * della matrice ad una nuova chiamata a Simmetrica.
 * Ad esempio, se N = 7 e la funzione viene chiamata con Dim=5, essa elabora
 * una sottomatrice 5 x 5 contenuta nella matrice Mat. Precisamente, la
 * funzione verifica se la matrice Mat e' del tipo
 *
 *    Y Y Y Y Y Y Y      0   indice delle righe
 *    Y Y Y Y Y Y Y      1
 *    Y Y 1 2 3 4 5      2 = N-Dim
 *    Y Y 2 X X X X      3
 *    Y Y 3 X X X X      4
 *    Y Y 4 X X X X      5
 *    Y Y 5 X X X X      6 = N-1
```

```
 *
 *    0 1 2 3 4 5 6  indice delle colonne
 *        |       |
 *        |       +--> N-1
 *        +--> N-Dim
 *
 * Se lo e', la funzione richiama se' stessa, con valori dei parametri
 * Mat e Dim-1. In questo modo, la nuova chiamata a Simmetrica elabora la
 * matrice i cui elementi sono indicati con X.
 */

void LeggiMatrice(Matrice Mat);
/* legge da tastiera il contenuto di una matrice N x N */

void StampaMatrice(Matrice Mat, int Dim);
/* riceve in ingresso una matrice di N x N elementi, di cui considera la
 * sottomatrice di Dim x Dim elementi ottenuta considerando i valori degli
 * indici che vanno da N-Dim a N-1, e stampa a schermo il contenuto della
 * sottomatrice */


int main()
{
  Matrice MatriceTest;

  LeggiMatrice(MatriceTest);

  if (true == Simmetrica(MatriceTest, N))
  {
    printf("\nLa matrice e' simmetrica.\n");
  }
  else
  {
    printf("\nLa matrice non e' simmetrica.\n");
  }

  return 0;
}


boolean Simmetrica(Matrice Mat, int Dim)
```

```
{
  int Indice;
  boolean Uguali;

#ifdef DEBUG
    printf("\nVa determinata la simmetria della matrice");
    StampaMatrice(Mat, Dim);
#endif

  if (1 == Dim)
  /* matrice composta da un solo elemento */
  {
    return true;
  }

  Uguali = true;

  /* comparazione della riga di indice (N-Dim) con la colonna di indice
   * (N-Dim), ovvero della prima riga e della prima colonna, della regione
   * di interesse dell'array Mat */
  for (Indice = N-Dim; Indice < N; ++Indice)
  {
    if (Mat[N-Dim][Indice] != Mat[Indice][N-Dim])
    {
      Uguali = false;
    }
  }

  if (false == Uguali)
  {
    return false;
  }
  else
  {
    return (Simmetrica (Mat, Dim-1));
  }
}


void LeggiMatrice(Matrice Mat)
{
```

```
  int ContR, ContC;

  printf("\n");
  for ( ContR = 0; ContR < N; ++ContR )
  {
    for ( ContC = 0; ContC < N; ++ContC )
    {
      printf("Valore elemento (%d, %d) della matrice?  ", ContR+1, ContC+1);
      scanf("%d", &Mat[ContR][ContC]);
    }
  }
}


void StampaMatrice(Matrice Mat, int Dim)
{
  int ContR, ContC;

  printf("\n");
  for ( ContR = N-Dim; ContR < N; ++ContR )
  {
    printf("\n");
    for ( ContC = N-Dim; ContC < N; ++ContC )
    {
      printf("\t");
      printf("%d", Mat[ContR][ContC]);
    }
  }

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