/* ____ *

Towers of Hanoi

This program gives an answer to the following famous problem (towers of Hanoi). There is a legend that when one of the temples in Hanoi was constructed, three poles were erected and a tower consisting of 64 golden discs was arranged on one pole, their sizes decreasing regularly from bottom to top. The monks were to move the tower of discs to the opposite pole, moving only one at a time, and never putting any size disc above a smaller one. The job was to be done in the minimum numbers of moves. What strategy for moving discs will accomplish this optimum transfer?

* ____ */

/* ____________________ hanoi.c ____________________ */

```
#include <stdio.h>
#define ARRAY 8                         /* disc の数                                        */

int disc[3][ARRAY];                     /* disc に関するデータの置き場所                    */

void init_array()                       /* disc に関するデータの初期化                      */
{
    int j;

    for (j = 0; j < ARRAY; ++j)         /* [問題] この for ループを抜け出した後、2 次元配列 */
                                        /* disc[][] にはどのようなデータが代入されることに  */
                                        /* なるか、具体的に述べよ。                         */
      {
          disc[0][j] = ARRAY - j;
          disc[1][j] = 0;
          disc[2][j] = 0;
      }
}

void print_result()                     /* 結果の表示                                       */
{
    static long counter = 0;            /* [問題] ここで static と宣言したのは、なぜなのか  */
                                        /* 考えよ。もしも static を削除したら、何が起こる   */
                                        /* か、実験をしてみよう。                           */

    int i, j;

    printf("---#%d---\n", ++counter);
    for (i = 0; i <= 2; ++i)
      {
          printf("[%d] ", i);
          for (j = 0; j < ARRAY; ++j)
            {
                if (disc[i][j] != 0)
                    printf("%d ", disc[i][j]);
```

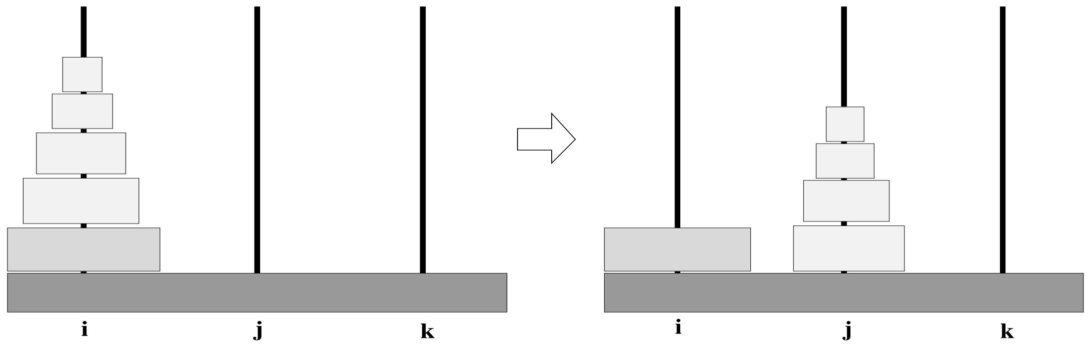
```
                else
                    break;
            }
        printf("\n");
      }
}

void move_one_disc(i, j)                /* 1枚の disc を pole i から pole j に移動する            */
{
    static int *ptr[3] = {disc[0] + ARRAY, disc[1], disc[2]};

    (*ptr[j]++) = (*--ptr[i]);          /* [問題] ++ はポインタの後にあり、-- はポインタ          */
                                        /* の前にある。なぜ、そのようにしなければならない         */
                                        /* のか説明せよ。                                         */
    *ptr[i] = 0;
}

void move_discs(n, i, j, k)             /* 上から n 枚目までの disc を、pole i から pole j に     */
                                        /* pole k を経由して、移動する                             */
int n, i, j, k;
{
    if (n >= 1)
      {
        move_discs(n - 1, i, k, j);     /* 関数 move_discs() の中で、さらに自分自身               */
        move_one_disc(i, j);            /* move_discs() が使われている。このような                */
        print_result();                 /* 手法は、「再帰的呼びだし」といわれる。一見             */
        move_discs(n - 1, k, j, i);     /* 複雑に見える問題でも、再帰的手法を用いると、           */
      }                                 /* 簡単に解けてしまうことが、しばしばある。               */
}

/*
```



たとえば、関数 move_discs(4,i,j,k) を呼び出すことは、上図のような操作をすることに対応する。 */

```
main()
{
    init_array();
    move_discs(ARRAY, 0, 1, 2);         /* ARRAY 枚の disc を pole 0 から pole 1 に pole 2        */
                                        /* を経由して、移動する                                   */
}
```