```
/* EE2310 Lab05. Permutations
107061112, 王昊文
Date: 2018.10.22
   107061112, 王昊文
   Date: 2018/10/22 // should be indented.
*/

#include <stdio.h>
#define N 7                     // macro N to determine the array

int main(void) {
    int i, j, k, m, a[N], temp, stay, counter = 1;
    /* i as a counter for a[N] to store numbers, it either
    counts up or counts down. j as a[j] is the one two
    swap with a[k], k as a[k], a[N] is the array, temp
    to help swap two arrays, stay is to determine when to
    jump out a loop, counter counts how many permutations
    are there. */
    // Comments can be indented one more level for better legibility.
    // Can insert a blank line here.
    printf("permutation #%d:", counter);    // first line
    for (i = 0; i < N; i++) {       // find out the first permutaion
                                                       permutaion // Spelling
        a[i] = i + 1;
        printf("%2d", a[i]);   // prints out number from small to big
    }
    printf("\n");
    /* if stay == 3, there are no more a[j] < a[j + 1], then
    all the permutations found */
    while (stay != 3) {
           // Is 'stay' initialized?
        for (i = N - 2, stay = 0, j = -1;  i >= 0 && stay == 0; i--) {
        /* let i count from the back and count down. the first
        a[i] < a[i + 1] is the biggest a[j]. Once found, stay == 1
        will jump out the for loop. we initialize j == -1, since
        it is impossible for j == -1, if no j found, the value remains
        and it's time to jump out the while loop */
            if (a[i] < a[i + 1]) {  //a[j] < a[j + 1]
                j = i;              // store in j
                stay = 1;           // jump out the loop
                counter++;          // one more permutation found
            }
        }
        if (j == -1)        // no more a[j] < a[j + 1] found
```

```
            stay = 3;       // jump out the while loop
        for (k = N - 1; k >= 0 && stay == 1; k--) {
        /* count k from the back, the first one to find is the
        biggest a[k] > a[j], then swap */
            if (a[k] > a[j]) {
                temp = a[k];
                a[k] = a[j];
                a[j] = temp;    // swap arrays
                stay = 2;       // jump to the next procedure
            }
        }
        if (stay != 3)          // if no j found then don't print
            printf("permutation #%d:", counter);
        for (i = j + 1, m = N - 1; i < m && stay == 2; i++, m--) {
        /* we want to swap from a[j + 1] to a[N - 1], let the
        a[j + 1] and a[N - 1] swap first, then a[j + 2] and a[N - 2}
        .... and so on, until the two arrays are the same
        or neighboring */
            temp = a[i];        // swap
            a[i] = a[m];
            a[m] = temp;
        }
        for (i = 0; i < N && stay!= 3; i++) { // print out permutaion
                                                           permutaion // Spelling
            printf("%2d", a[i]);
        }
        if (stay != 3) {        // if no j found don't print
            printf("\n");
        }
    }
    // last line
    printf("  Total number of permutations is %d\n", counter);
        return 0;
    return 0;
}
```

// Program can be wrong due to an uninit variable.
// Program logic can be simplified.
Score: 80