lab05

```
$ gcc lab05.c

$ a.out
permutation #1: 1 2 3 4 5 6 7
permutation #2: 1 2 3 4 5 7 6
permutation #3: 1 2 3 4 6 5 7
.....
permutation #5039: 7 6 5 4 3 1 2
permutation #5040: 7 6 5 4 3 2 1
  Total number of permutations is 5040
```

---

score: 81.0
o. [Output] Program output is correct, good.
o. [Coding] lab05.c spelling errors: decresingly(1), determinated(1), storaged(2)
o. [Array] Ans is not needed.
o. [Efficiency] can be improved.

lab05.c

```c
// EE231002 Lab05. Permutations
// 110060007, 黃俊穎
// 2021/11/08

#include <stdio.h>                  // I/O library
#define N 7                         // question setting number
#define true 1                      // set true value equal to 1
#define false 0                     // set false value equal to 0

int main(void)                      // start the main function
{
    int A[N], Ans[N];               // array for answer and storage
    int i, j, k, t;                 // variables for loops and string
    int arr_j, arr_k;               // number in given array
    int index_j, index_k;           // number of given array
    int total_num = 1;              // initialize number of factorial
    int saver;                      // storage for swapping number
    int all_index;                  // value of string number
    int T_F = true;
    // a determinated value to decide if the string still run in loops
    int counter = 1;                // count for permutation number

    // initialize array, storage array and value of factorial
    for (i = 0; i < N; i++) {
        A[i] = i + 1;
        Ans[i] = i + 1;
        total_num *= (i + 1);
    }

    // show sequence for printing
    while (T_F) {
        // set step 1 to be terminated condition
        T_F = false;
        printf("permutation #%d:", counter++);

        // show the result of every sequence
        for (i = 0; i < N; i++) {
            printf(" %d", Ans[i]);
        }
        printf("\n");               // skip to next line
```

```
41          all_index = N;              // save total number in an array
42
43          // step 1: find the largest index such that A[j] < A[j + 1]
44          for (j = 0; j < N - 1; j++) {
45              if (A[j] < A[j + 1]) {
46                  arr_j = A[j];       // save the value in A[j]
47                  index_j = j;        // save j
48                  T_F = true;         // continue following command
49              }
50          }
51
52          // step 2: find the largest index k such that A[j] < A[k]
53          for (k = index_j; k < N; k++) {
54              if (arr_j < A[k]) {
55                  arr_k = A[k];       // save the value in A[k]
56                  index_k = k;        // save k
57              }
58          }
59          // step 3: swap A[j] with A[k]
60          saver = arr_k;
61          arr_k = arr_j;
62          arr_j = saver;
63          A[index_j] = arr_j;
64          A[index_k] = arr_k;
65
66          // step 4: reverse the sequence from A[j + 1] up to and
67          // including the last element A[N - 1]
68          for (i = 0; i <= index_j; i++) {
69              Ans[i] = A[i];
70              // fill in first part value in storaged array
71          }
72          for (i = index_j + 1; i < N; i++) {
73              Ans[i] = A[all_index - 1];
74              // fill in second part value in storaged array
75              all_index--;
76              // fill in corresponding index decresingly
77          }
78          // reverse storage to original array to do next permutation
79          for (i = 0; i < N; i++) {
80              A[i] = Ans[i];          // redo for next permutation
81          }
```

```
82      }
83      // print out the result
84      printf("  Total number of permutations is %d\n", total_num);
85      return 0;                       // finish the main function
86  }
```