lab06

```
$ gcc lab06.c

$ ./a.out
permutation #1: A B C D E F G
permutation #2: A B C D E G F
permutation #3: A B C D F E G
permutation #4: A B C D F G E
permutation #5: A B C D G E F
permutation #6: A B C D G F E
.....
permutation #5036: G F E D A C B
permutation #5037: G F E D B A C
permutation #5038: G F E D B C A
permutation #5039: G F E D C A B
permutation #5040: G F E D C B A
  Total number of permutations is 5040
```

```
score: 91.0
o. [Output] Program output is correct, good.
o. [Array] 'A' should store char data.
o. [Efficiency] can still be improved.
o. [Format] Program format can be improved.
```

lab06.c

```c
// EE231002 Lab06. Permutations
// 111060023, 黃柏霖
// 2022/10/25

#include <stdio.h>
#define N 7

int main(void)
{
    typedef int bool;               // define a type bool
```
typedef should be global
```c
    char A[N];                      // a set of distinct alphabets
    char tmp;                       // temporary memory for chars
    int i;                          // loop controller
    int j;                          // the largest index that A[j] < A[j + 1]
    int k;                          // the largest index that A[j] < A[k]
    int count = 1;                  // count how many set
    bool go = 1;                    // keep going if j is found
    bool stop;                      // stop searching k if k is found

    printf("permutation #%d:", count);      // imply which set is print now
    for (i = 0; i < N; i++)                 // initialize and print the 1st set
        printf(" %c", A[i] = 'A' + i);      // initialize the element and print
    printf("\n");                           // end line
    while (go) {                            // start permuting
        go = 0;                             // default that j is not found
        for (i = 0; i < N - 1; i++) {       // finding j from A[0] to A[N - 1]
            if (A[i] < A[i + 1]) {          // finding j
                j = i;                      // store j
                go = 1;                     // j is found
            }
        }
        if (go == 1) {                      // do the following things if go = 1
            count++;                        // one more set is found
            stop = 0;                       // k still not found, keep searching
            // keep finding k until it's found
            for (i = N - 1; i > j && stop != 1; i--) {
                if (A[i] > A[j]) {          // finding k
                    k = i;                  // store k
                    stop = 1;               // k is found, stop searching
                }
            }
            tmp = A[j];                     // store A[j] in tmp
            A[j] = A[k];                    // change A[j] with A[k]
            A[k] = tmp;                     // change A[k] with tmp, swap done
            // keep swapping from j + 1 to the mid of j + 1 and N - 1
            for (i = j + 1; i <= (N + j) / 2; i++) {
                tmp = A[N + j - i];         // store A[N + j - i] in tmp
                A[N + j - i] = A[i];        // change A[N + j - i] with A[i]
                A[i] = tmp;                 // change A[i] with tmp, swap done
            }
            printf("permutation #%d:",
                count);                     // imply which set is print now
            for (i = 0; i < N; i++)         // print set
                printf(" %c", A[i]);        // print the ith element of set
            printf("\n");                   // end line
        }
    }
    printf("  Total number of permutations is %d\n",
        count);                             // print the total #set
    return 0;
}
```