```c
/*  EE231002 Lab05. Permutationsi
                    Permutationsi // Spelling
    107061113, 李柏葳
    Date:2018/10/22
*/

#include <stdio.h>
#define N 7

int main(void) {
    int j;  // set to store largest index which permu[i] < permu[i-1]
    int k;  // to check largrst index which permu[j] <permu[k]
                        largrst // Spelling
    int i;  // used to index permu[]
    int out = 0;    // to get out the loop if not find j
    int permu[N-1] = {0};   // to store permutation's numbers
    int permu[N-1] // Why dimension N-1?
    int temp = 0;   // to temporary store number to switch
    int rev[N-1];   // to temperary store array number to switch
                          temperary // Spelling
    int num = 1;    // to count # number
    int exit = 0;   // to exit the loop

    for (i = 0; i < N; i++) {   // to loop setting process
        permu[i] = i + 1;   // to set number into array
    }
    while (exit == 0 && N >= 3 ) {  // to decide when to stop finding process
        printf("permutation #%d:", num);    // print the # at front of number
        num++;  // to count #
            for(i = 0; i < N  ; i++) {  // to print the permutation after :
        for (i = 0; i < N; i++) {  // indentation and space
            printf("%2d", permu[i]);
        }
        printf("\n");   // go to next line
            for (i = N-1; out == 0; i--) {  // to loop searching j process
                if (permu[i] > permu [i-1] && i > 0) {  // to find largest
                // index which permu[i] < permu[i - 1]
                    j = i - 1;  // store number which find from upper line
                    out++;  // bail out searching j process
                }
                if (i == 0){    // if can not find j
                    out++;  // bail out searching j process
                    exit++; //leave this entire searching loop
                }
```

```
40              }
41          out = 0;    // reset out value
42          for (i = N-1; out == 0; i--) {
43          // find largest index k if permu[k] > permu[j]
44              if (permu[i] > permu[j]) {
45                  k = i;
46                  out++;
47              }
48          }
49          out = 0;    // reset out value
50          temp = permu[j];    // to store permu[j] in temporary
51          permu[j] = permu[k];    // replace permu[j] with permu[k]
52          permu[k] = temp;    // replace permu[k] with permu[j]
53          for (i = j + 1; i <= N - 1 ; i++) {
54          // to store reverse number in another array
55              rev[i] = permu[i];
56          }
57          for (i = N - 1; i >= j + 1; i--) {
58              permu[i] = rev[j + N - i];  // put number in reversed permu array
59          }
60      }
61      if (N == 2) {   /* when to do if Given number <= 2
62      ( since the program above can only detect integer larger than 2
63        so I think it is easier just to print permutations than modify
64        the searching loops ) */
65          printf("permutation #1: 1 2\n"
66                 "permutation #2: 2 1\n");
67          num = N + 1;    // set total permutation number to 2
68      }
69      else if (N == 1) {
70          printf("permutation #1: 1\n");
71          num = N + 1;    // set permutation number to 1
72      }
73      else if (N <= 0) {  // to prevent user accdiently put in wrong number
                                                accdiently // Spelling
74          printf("Error!Number must larger than 0");
75          return 0;
76      }
77      printf("  Total number of permutations is %d\n", num - 1);
78      // print total number
79
80      return 0;
81  }
82
```

```
83
```

// Program execution can fail.
// Program format needs improvements (indentation and space).
Score: 63