Implement Johnson Trotter algorithm to generate permutations.

```
#include <stdio.h>
#include <stdlib.h>

#define LEFT -1
#define RIGHT 1

int n;
int perm[10];     // Current permutation
int dir[10];      // Direction of each element

void print_perm() {
   for (int i = 0; i < n; i++) {
      printf("%d ", perm[i]);
   }
   printf("\n");
}

// Find the largest mobile integer
int get_mobile() {
   int mobile = 0;
   int mobile_index = -1;

   for (int i = 0; i < n; i++) {
      int next = i + dir[i];
      if (next >= 0 && next < n && perm[i] > perm[next]) {
         if (perm[i] > mobile) {
            mobile = perm[i];
            mobile_index = i;
```

```
            }
        }
    }

    return mobile_index;
}

// Swap elements and their directions
void swap(int i, int j) {
    int temp = perm[i];
    perm[i] = perm[j];
    perm[j] = temp;

    temp = dir[i];
    dir[i] = dir[j];
    dir[j] = temp;
}

void generate_permutations() {
    print_perm();  // First permutation

    while (1) {
        int mobile_index = get_mobile();
        if (mobile_index == -1)
            break;

        int next = mobile_index + dir[mobile_index];
        swap(mobile_index, next);

        // Reverse direction of all elements greater than current
        for (int i = 0; i < n; i++) {
```

```
            if (perm[i] > perm[next]) {
                dir[i] = -dir[i];
            }
        }

        print_perm();
    }
}

int main() {
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    if (n <= 0 || n > 10) {
        printf("Enter a number between 1 and 10.\n");
        return 1;
    }

    // Initialize the permutation and direction
    for (int i = 0; i < n; i++) {
        perm[i] = i + 1;
        dir[i] = LEFT;
    }

    printf("Permutations using Johnson–Trotter Algorithm:\n");
    generate_permutations();

    return 0;
}
```

```
Output
Enter the number of elements: 2
Permutations using Johnson-Trotter Algorithm:
1 2
2 1
```