Implement Johnson Trotter algorithm to generate permutations.

Code:

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

#define LEFT -1
#define RIGHT 1

// Function to print the current permutation
void printPermutation(int* perm, int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", perm[i]);
    }
    printf("\n");
}

// Function to find the largest mobile integer
int getMobile(int* perm, int* dir, int n) {
    int mobile = 0;

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

    return mobile;
}

// Find the position of the given element
int findPosition(int* perm, int n, int mobile) {
    for (int i = 0; i < n; i++) {
        if (perm[i] == mobile)
            return i;
    }
    return -1;
}

void generatePermutations(int n) {
```

```
    int* perm = (int*)malloc(n * sizeof(int));
    int* dir = (int*)malloc(n * sizeof(int));

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

    int total = 1;
    for (int i = 2; i <= n; i++)
        total *= i;

    // Print first permutation
    printPermutation(perm, n);

    for (int count = 1; count < total; count++) {
        int mobile = getMobile(perm, dir, n);
        int pos = findPosition(perm, n, mobile);

        int swapWith = pos + dir[pos];

        // Swap values
        int temp = perm[pos];
        perm[pos] = perm[swapWith];
        perm[swapWith] = temp;

        // Swap directions too
        temp = dir[pos];
        dir[pos] = dir[swapWith];
        dir[swapWith] = temp;

        // Reverse direction of all elements greater than mobile
        for (int i = 0; i < n; i++) {
            if (perm[i] > mobile)
                dir[i] = -dir[i];
        }

        printPermutation(perm, n);
    }

    free(perm);
    free(dir);
}
```

```
int main() {
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    generatePermutations(n);
    return 0;
}
```

Output: