JOHNSON TROTTER

```
#include <stdio.h>
#include <stdbool.h>
#define RIGHT_TO_LEFT 0
#define LEFT_TO_RIGHT 1
int searchArr(int a[], int n, int mobile)
{
   for (int i = 0; i < n; i++)
   {
      if (a[i] == mobile)
         return i + 1;
   }
   return -1;
}
int getMobile(int a[], bool dir[], int n)
{
   int mobile_prev = 0, mobile = 0;
   for (int i = 0; i < n; i++)
   {
      if (dir[a[i] - 1] == RIGHT_TO_LEFT && i != 0)
      {
         if (a[i] > a[i - 1] && a[i] > mobile_prev)
         {
            mobile = a[i];
            mobile_prev = mobile;
         }
      }

      if (dir[a[i] - 1] == LEFT_TO_RIGHT && i != n - 1)
      {
         if (a[i] > a[i + 1] && a[i] > mobile_prev)
         {
            mobile = a[i];
            mobile_prev = mobile;
         }
      }
   }

   if (mobile == 0 && mobile_prev == 0)
      return 0;
   else
      return mobile;
```

```
}
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printOnePerm(int a[], bool dir[], int n)
{
    int mobile = getMobile(a, dir, n);
    int pos = searchArr(a, n, mobile);

    if (pos != -1)
    {
        if (dir[a[pos - 1] - 1] == RIGHT_TO_LEFT)
            swap(&a[pos - 1], &a[pos - 2]);
        else if (dir[a[pos - 1] - 1] == LEFT_TO_RIGHT)
            swap(&a[pos], &a[pos - 1]);
    }

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

int fact(int n)
{
    int res = 1;
    for (int i = 1; i <= n; i++)
        res = res * i;
    return res;
}
void printPermutation(int n)
{
    int a[n];
    bool dir[n];

    for (int i = 0; i < n; i++)
    {
        a[i] = i + 1;
        printf("%d ", a[i]);
    }
```

```
    printf("\n");

    for (int i = 0; i < n; i++)
        dir[i] = RIGHT_TO_LEFT;

    for (int i = 1; i < fact(n); i++)
        printOnePerm(a, dir, n);
}
int main()
{
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    printPermutation(n);
    return 0;
}
```

```
Enter the value of n: 4
1 2 3 4
1 2 4 3
1 4 2 3
4 1 2 3
4 1 3 2
4 3 1 2
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
```