

OUTPUT FOR JOHNSON TROTTER

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 bool LEFT_TO_RIGHT = true;
4 bool RIGHT_TO_LEFT = false;
5 void swap(int *a, int *b) {
6     int temp = *a;
7     *a = *b;
8     *b = temp;
9 }
10 int searchArr(int a[], int n, int mobile) {
11     for (int i = 0; i < n; i++) {
12         if (a[i] == mobile) {
13             return i + 1;
14         }
15     }
16 }
17 int getMobile(int a[], bool dir[], int n) {
18     int mobile_prev = 0, mobile = 0;
19     for (int i = 0; i < n; i++) {
20         if (dir[a[i] - 1] == RIGHT_TO_LEFT && i != 0)
21             if (a[i] > a[i - 1] && a[i] > mobile_prev)
22                 mobile = a[i];
23         else if (dir[a[i] - 1] == LEFT_TO_RIGHT && i != n - 1)
24             if (a[i] > a[i + 1] && a[i] > mobile_prev)
25                 mobile = a[i];
26         mobile_prev = mobile;
27     }
28     return mobile;
29 }
30 int printOnePerm(int a[], bool dir[], int n) {
31     int mobile = getMobile(a, dir, n);
32     printf("%d ", mobile);
33     if (n % 10 == 0) printf("\n");
34 }
35 void printPermutation(int n) {
36     int a[n];
37     bool dir[n];
38     for (int i = 0; i < n; i++) {
39         a[i] = i + 1;
40         dir[i] = LEFT_TO_RIGHT;
41     }
42     printOnePerm(a, dir, n);
43     for (int i = 0; i < n; i++) {
44         int pos = searchArr(a, n, mobile);
45         if (dir[pos - 1] == RIGHT_TO_LEFT)
46             swap(&a[pos - 1], &a[pos - 2]);
47         else if (dir[pos - 1] == LEFT_TO_RIGHT)
48             swap(&a[pos], &a[pos - 1]);
49         dir[pos - 1] = !dir[pos - 1];
50         for (int i = 0; i < n; i++) {
51             if (a[i] > mobile)
52                 if (dir[a[i] - 1] == LEFT_TO_RIGHT)
53                     dir[a[i] - 1] = RIGHT_TO_LEFT;
54                 else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
55                     dir[a[i] - 1] = LEFT_TO_RIGHT;
56             else if (dir[a[i] - 1] == LEFT_TO_RIGHT)
57                 dir[a[i] - 1] = LEFT_TO_RIGHT;
58             else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
59                 dir[a[i] - 1] = RIGHT_TO_LEFT;
60         }
61         printOnePerm(a, dir, n);
62     }
63 }
64 int fact(int n) {
65     int res = 1;
66     for (int i = 1; i <= n; i++)
67         res = res * i;
68     return res;
69 }
70 void printPermutation(int n) {
71     int a[n];
72     bool dir[n];
73     for (int i = 0; i < n; i++) {
74         a[i] = i + 1;
75         dir[i] = LEFT_TO_RIGHT;
76     }
77     printOnePerm(a, dir, n);
78     for (int i = 0; i < n; i++) {
79         int pos = searchArr(a, n, mobile);
80         if (dir[pos - 1] == RIGHT_TO_LEFT)
81             swap(&a[pos - 1], &a[pos - 2]);
82         else if (dir[pos - 1] == LEFT_TO_RIGHT)
83             swap(&a[pos], &a[pos - 1]);
84         dir[pos - 1] = !dir[pos - 1];
85         for (int i = 0; i < n; i++) {
86             if (a[i] > mobile)
87                 if (dir[a[i] - 1] == LEFT_TO_RIGHT)
88                     dir[a[i] - 1] = RIGHT_TO_LEFT;
89                 else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
90                     dir[a[i] - 1] = LEFT_TO_RIGHT;
91             else if (dir[a[i] - 1] == LEFT_TO_RIGHT)
92                 dir[a[i] - 1] = LEFT_TO_RIGHT;
93             else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
94                 dir[a[i] - 1] = RIGHT_TO_LEFT;
95         }
96         printOnePerm(a, dir, n);
97     }
98 }
99 int main() {
100     int n;
101     printf("Enter the value of n: ");
102     scanf("%d", &n);
103     printf("All the permutations are :\n");
104     printPermutation(n);
105 }
```

The first screenshot shows the OnlineGDB IDE with the following C code:

```
main.c swap i
49 swap(a[pos], a[pos - 1]);
50 for (int i = 0; i < n; i++)
51 {
52     if (a[i] > mobile)
53     {
54         if (dir[a[i] - 1] == LEFT_TO_RIGHT)
55             dir[a[i] - 1] = RIGHT_TO_LEFT;
56         else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
57             dir[a[i] - 1] = LEFT_TO_RIGHT;
58     }
59 }
60 for (int i = 0; i < n; i++)
61     printf("%d", a[i]);
62     printf(" ");
63 }
64 int fact(int n) {
65     int res = 1;
66     for (int i = 1; i <= n; i++)
67         res = res * i;
68     return res;
69 }
70 void printPermutation(int n) {
71     int a[n];
72     bool dir[n];
73     for (int i = 0; i < n; i++)
74     {
75         a[i] = i + 1;
76         printf("%d", a[i]);
77     }
78     printf(" ");
79     for (int i = 0; i < n; i++)
80         dir[i] = RIGHT_TO_LEFT;
81     for (int i = 1; i < fact(n); i++)
82         printOnePerm(a, dir, n);
83 }
84 int main() {
85     int n;
86     printf("Enter the value of n: ");
87     scanf("%d", &n);
88     printf("All the permutations are : \n");
89     printPermutation(n);
90     printf("\n");
91     return 0;
92 }
```

The second screenshot shows the output of the program for n=4:

```
Enter the value of n: 4
All the permutations are :
1234 1243 1423 4123 4132 1432 1324 3124 3142 3412 4312 4321 3421 3241 3214 2314 2341 4231 4213 2413 2143 2134
...Program finished with exit code 0
Press ENTER to exit console.
```

CODE FOR JOHNSON TROTTER

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

```
#include<stdbool.h>
```

```
bool LEFT_TO_RIGHT = true;
```

```
bool RIGHT_TO_LEFT = false;
```

```

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int searchArr(int a[], int n, int mobile) {
    for (int i = 0; i < n; i++) {
        if (a[i] == mobile) {
            return i + 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;
}

int printOnePerm(int a[], bool dir[], int n) {
int mobile = getMobile(a, dir, n);
int pos = searchArr(a, n, mobile);
if (dir[a[pos] - 1] == RIGHT_TO_LEFT)
swap(&a[pos - 1], &a[pos - 2]);
else if (dir[a[pos] - 1] == LEFT_TO_RIGHT)
swap(&a[pos], &a[pos - 1]);
for (int i = 0; i < n; i++)
{
if (a[i] > mobile)
{
if (dir[a[i] - 1] == LEFT_TO_RIGHT)
dir[a[i] - 1] = RIGHT_TO_LEFT;
else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
dir[a[i] - 1] = LEFT_TO_RIGHT;
}
}
for (int i = 0; i < n; i++)
printf("%d", a[i]);
printf(" ");
}

```

```

}

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(" ");
    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);
    printf("All the permutations are : \n");
    printPermutation(n);
    printf("\n");
    return 0;}

```