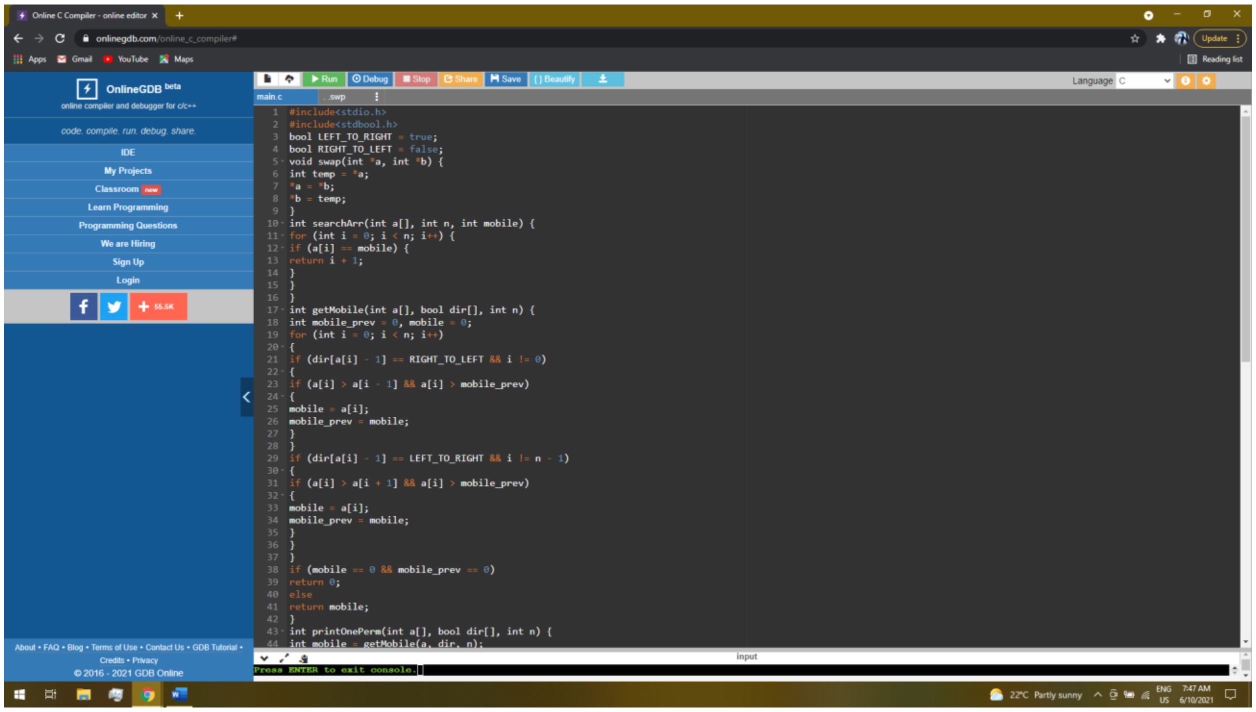
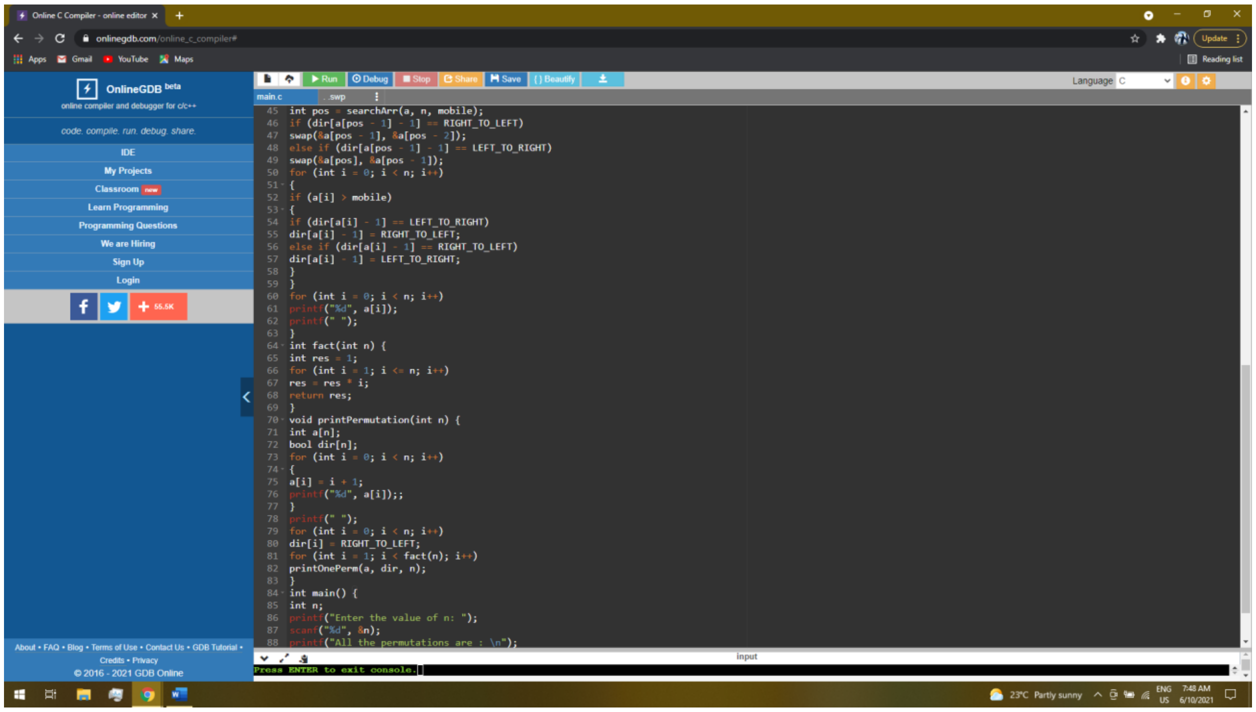
OUTPUT FOR JOHNSON TROTTER



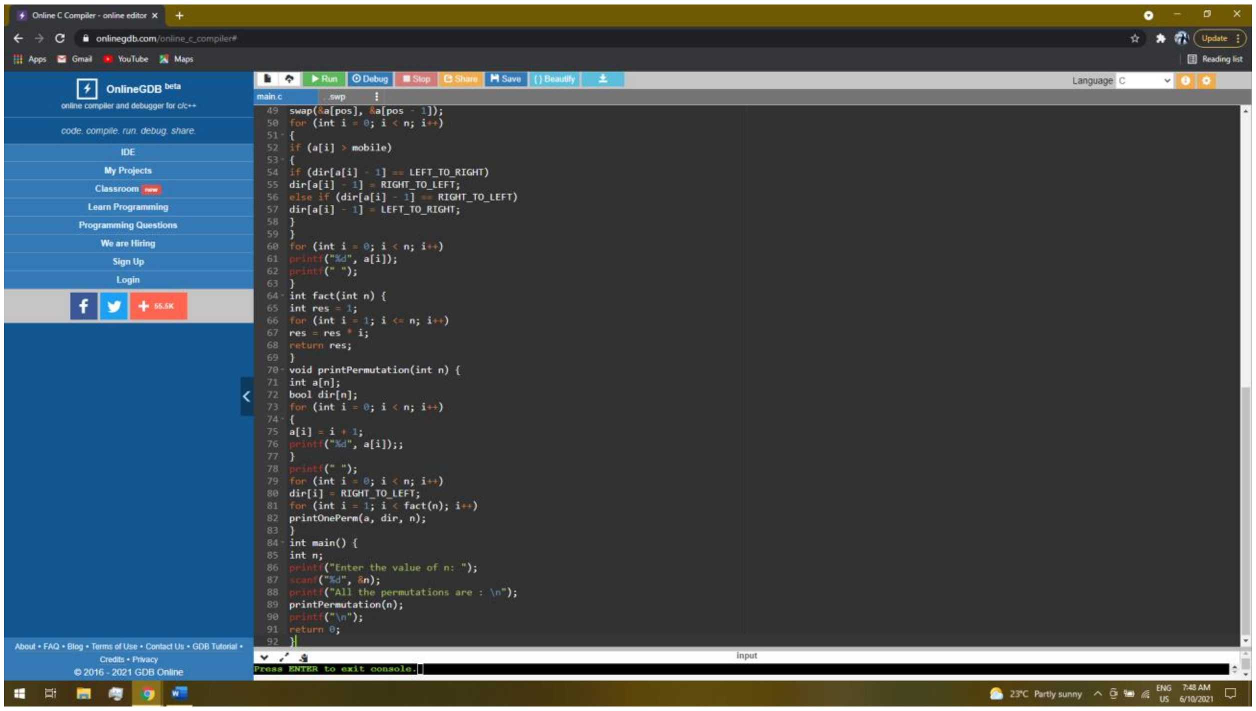
```c
#include<stdio.h>
#include<stdbool.h>
bool LEFT_TO_RIGHT = true;
bool RIGHT_TO_LEFT = false;
void swap(int *a, int *b) {
int temp = *a;
*a = *b;
*b = temp;
}
int searchArr(int a[], int n, int mobile) {
for (int i = 0; i < n; i++) {
if (a[i] == mobile) {
return i + 1;
}
}
}
int getMobile(int a[], 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);
```

```
Press ENTER to exit console.
```



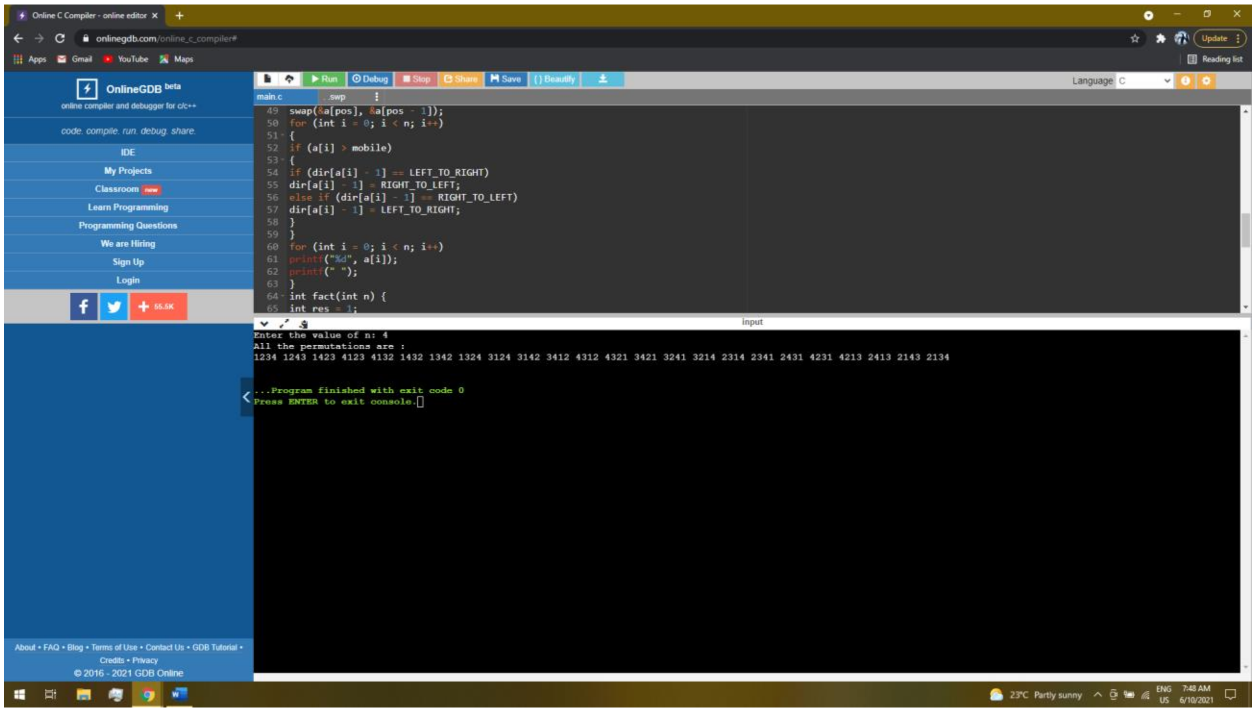
```c
int pos = searchArr(a, n, mobile);
if (dir[a[pos - 1] - 1] == RIGHT_TO_LEFT)
swap(&a[pos - 1], &a[pos - 2]);
else if (dir[a[pos - 1] - 1] == LEFT_TO_RIGHT)
swap(&a[pos], &a[pos - 1]);
for (int i = 0; i < n; i++)
{
if (a[i] > mobile)
{
if (dir[a[i] - 1] == LEFT_TO_RIGHT)
dir[a[i] - 1] = RIGHT_TO_LEFT;
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");
```

```
Press ENTER to exit console.
```



```c
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;
}
```

```
Press ENTER to exit console.
```



```c
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;
```

```
Enter the value of n: 4
All the permutations are :
1234 1243 1423 4123 4132 1432 1342 1324 3124 3142 3412 4312 4321 3421 3241 3214 2314 2341 2431 4231 4213 2413 2143 2134


...Program finished with exit code 0
Press ENTER to exit console.
```

CODE FOR JOHNSON TROTTER

```c
#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] - 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++)
{
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;}
```