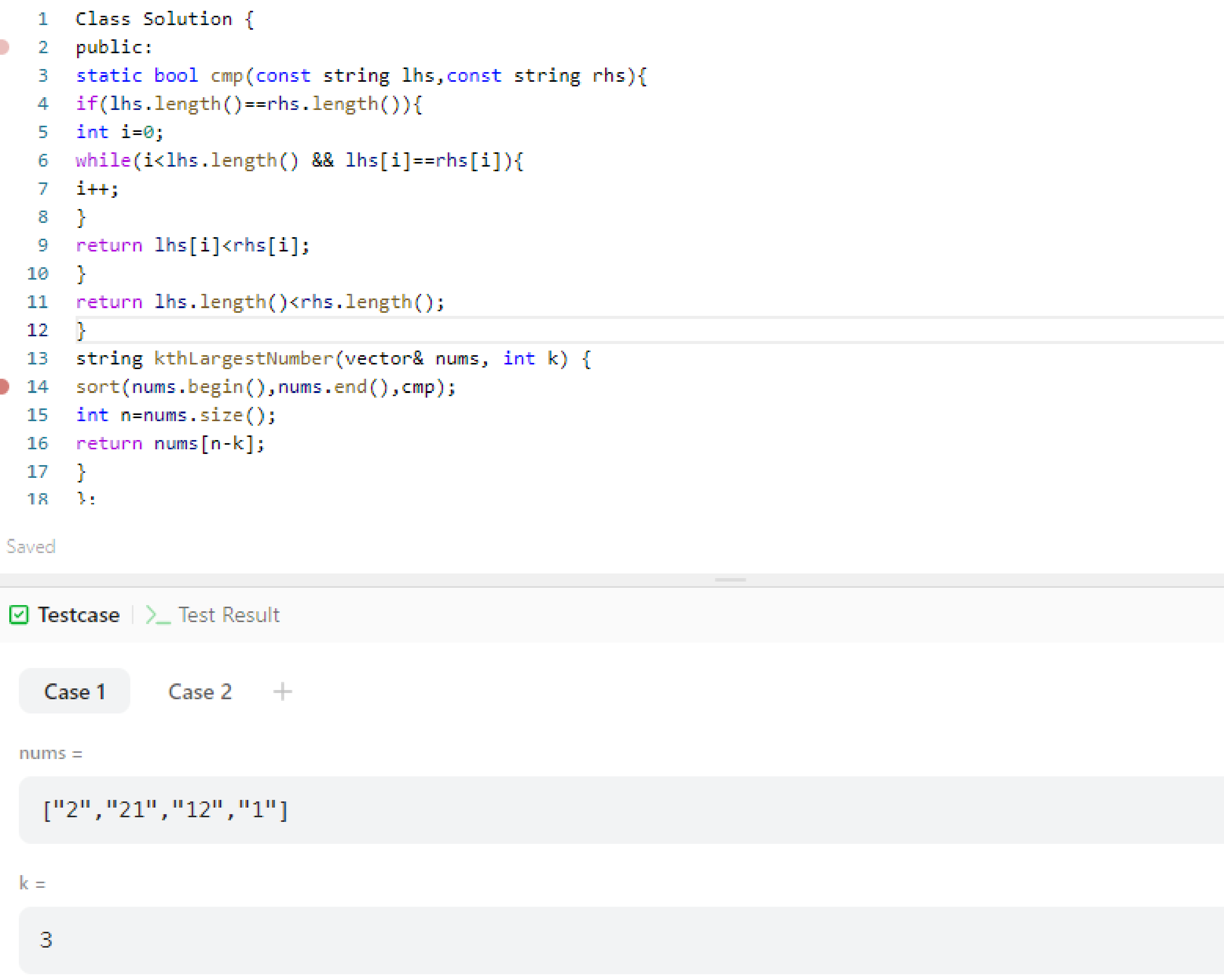


```
Class Solution {
public:
static bool cmp(const string lhs,const string rhs){
if(lhs.length()==rhs.length()){
int i=0;
while(i<lhs.length() && lhs[i]==rhs[i]){
i++;
}
return lhs[i]<rhs[i];
}
return lhs.length()<rhs.length();
}
string kthLargestNumber(vector& nums, int k) {
sort(nums.begin(),nums.end(),cmp);
int n=nums.size();
return nums[n-k];
}
};
```

Saved

Testcase | Test Result

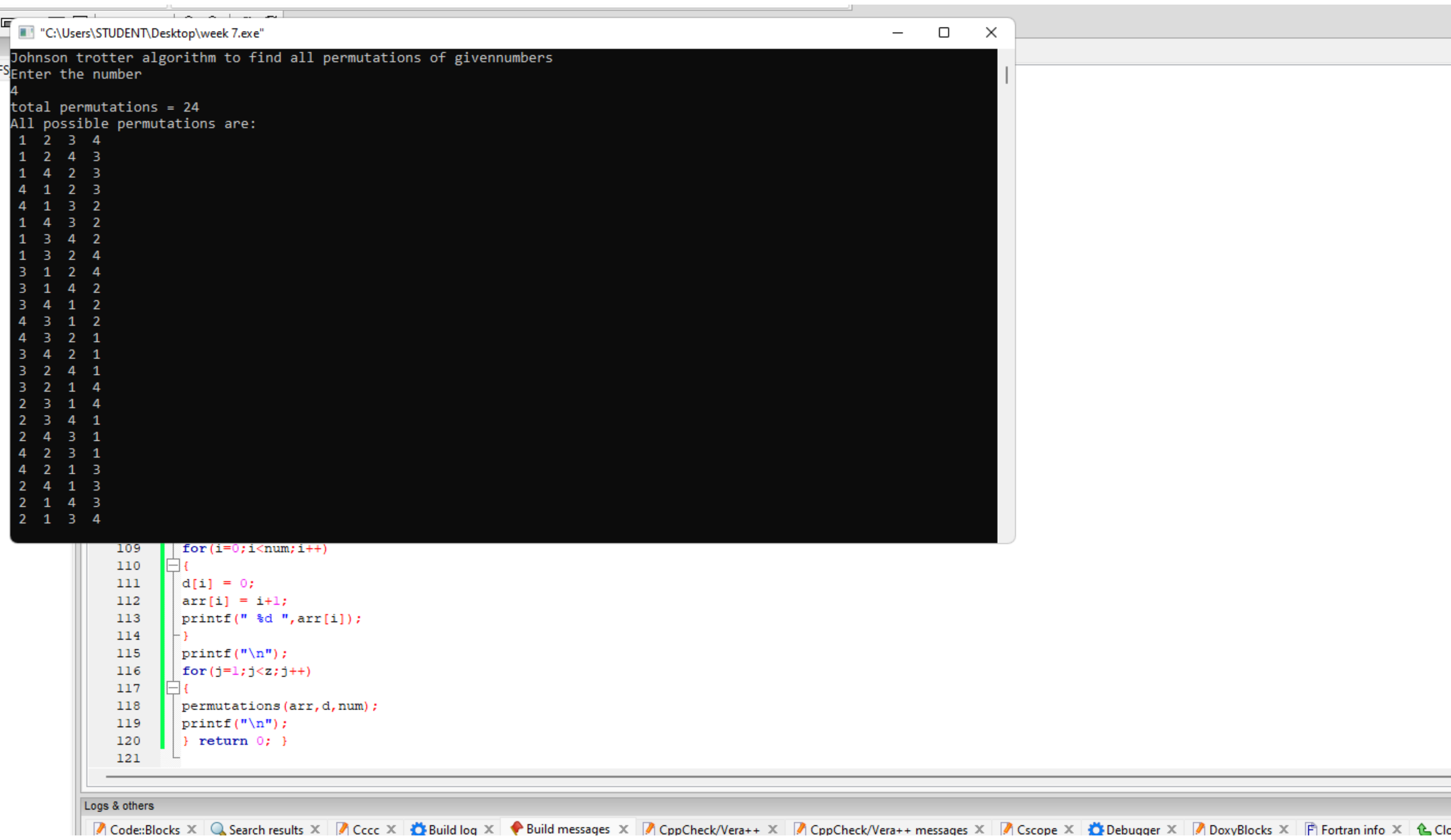
Case 1 Case 2 +

nums =

```
["2","21","12","1"]
```

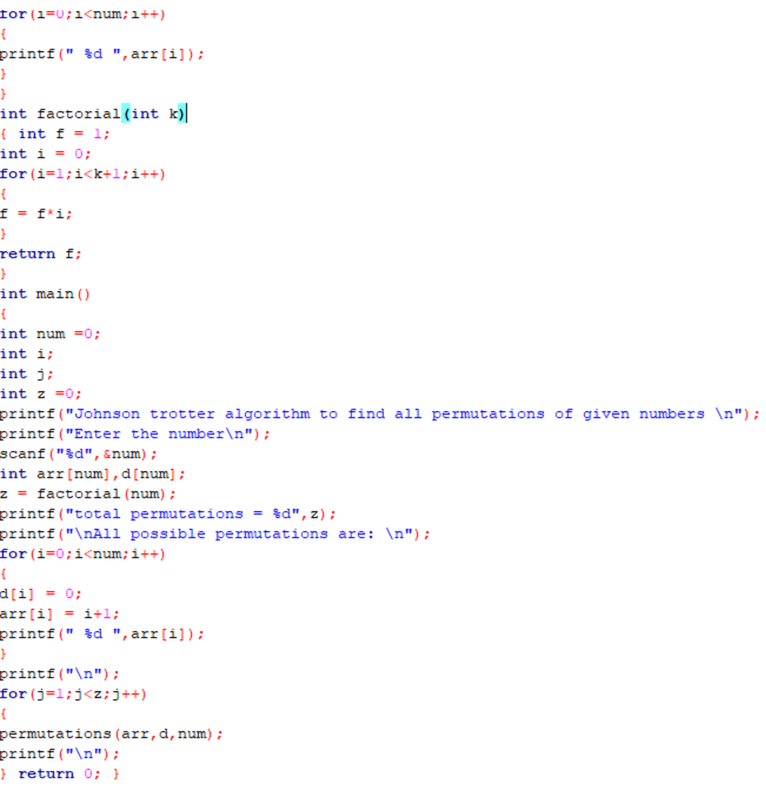
k =

```
3
```

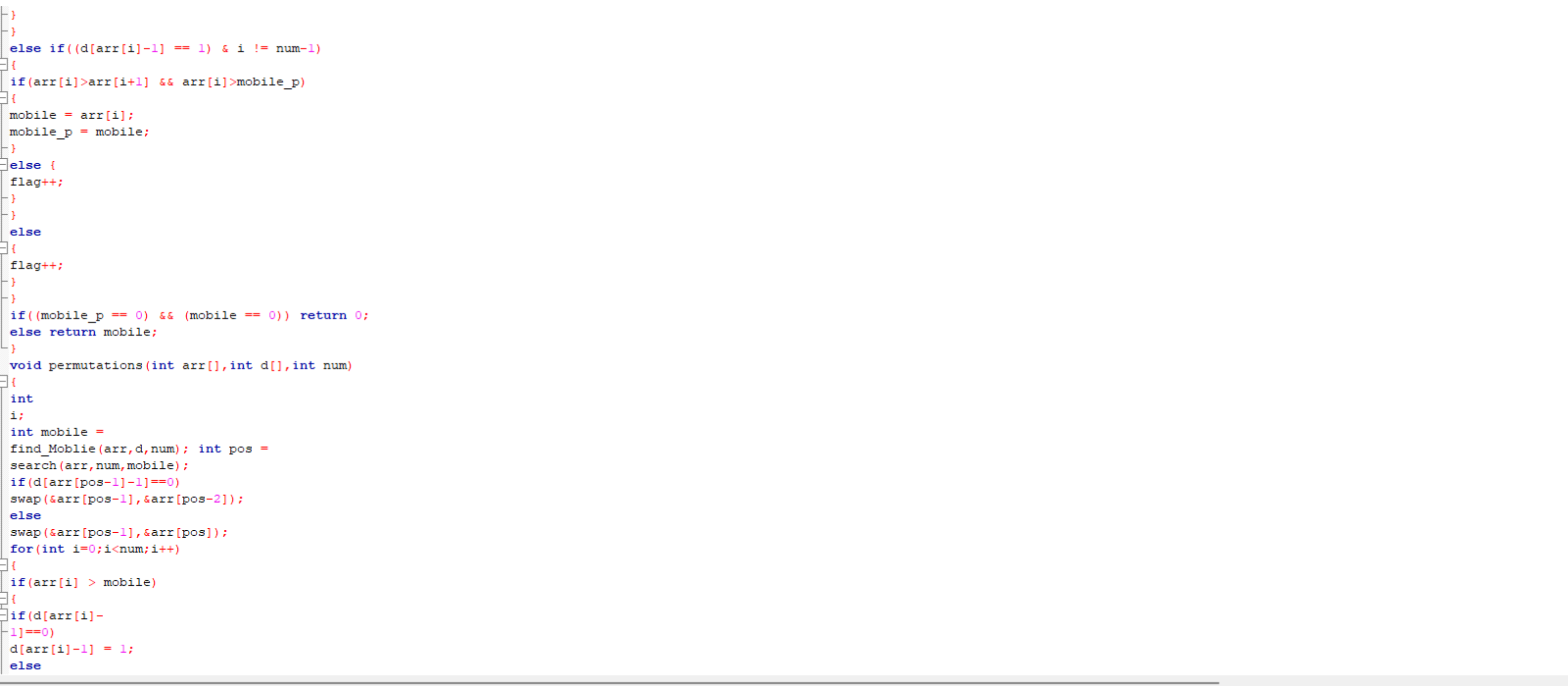


```
Johnson trotter algorithm to find all permutations of givennumbers
Enter the number
4
total permutations = 24
All possible permutations are:
 1  2  3  4
 1  2  4  3
 1  4  2  3
 4  1  2  3
 4  1  3  2
 1  4  3  2
 1  3  4  2
 1  3  2  4
 3  1  2  4
 3  1  4  2
 3  4  1  2
 4  3  1  2
 4  3  2  1
 3  4  2  1
 3  2  4  1
 3  2  1  4
 2  3  1  4
 2  3  4  1
 2  4  3  1
 4  2  3  1
 4  2  1  3
 2  4  1  3
 2  1  4  3
 2  1  3  4
```

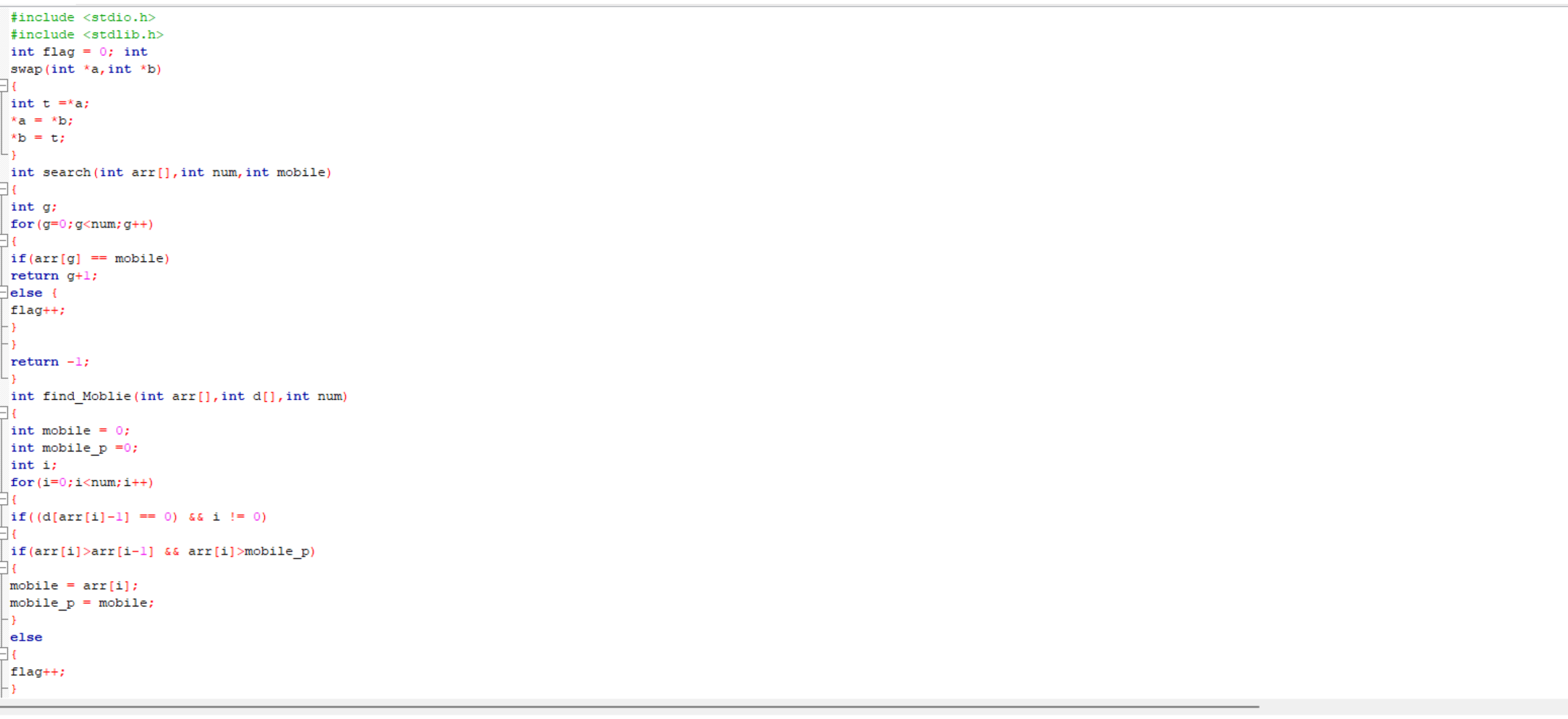
```
for(i=0;i<num;i++)
{
d[i] = 0;
arr[i] = i+1;
printf(" %d ",arr[i]);
}
printf("\n");
for(j=1;j<z;j++)
{
permutations(arr,d,num);
printf("\n");
} return 0; }
```



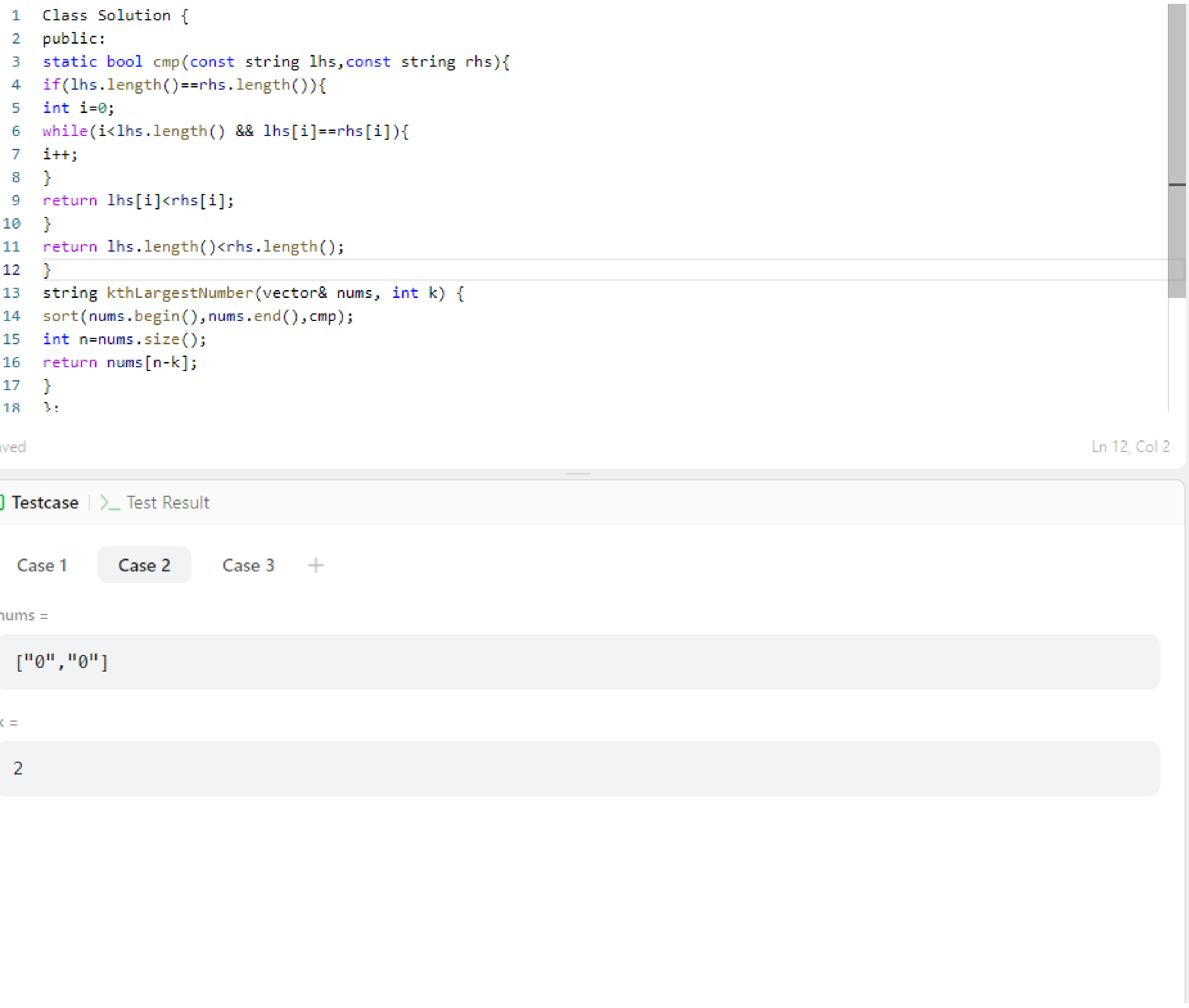
```
for(i=0;i<num;i++)
{
printf(" %d ",arr[i]);
}
}
int factorial(int k)
{ int f = 1;
int i = 0;
for(i=1;i<k+1;i++)
{
f = f*i;
}
return f;
}
int main()
{
int num =0;
int i;
int j;
int z =0;
printf("Johnson trotter algorithm to find all permutations of given numbers \n");
printf("Enter the number\n");
scanf("%d",&num);
int arr[num],d[num];
z = factorial(num);
printf("total permutations = %d",z);
printf("\nAll possible permutations are: \n");
for(i=0;i<num;i++)
{
d[i] = 0;
arr[i] = i+1;
printf(" %d ",arr[i]);
}
printf("\n");
for(j=1;j<z;j++)
{
permutations(arr,d,num);
printf("\n");
} return 0; }
```



```
}
}
else if((d[arr[i]-1] == 1) & i != num-1)
{
if(arr[i]>arr[i+1] && arr[i]>mobile_p)
{
mobile = arr[i];
mobile_p = mobile;
}
else {
flag++;
}
}
else
{
flag++;
}
}
if((mobile_p == 0) && (mobile == 0)) return 0;
else return mobile;
}
void permutations(int arr[],int d[],int num)
{
int
i;
int mobile =
find_Moblie(arr,d,num); int pos =
search(arr,num,mobile);
if(d[arr[pos-1]-1]==0)
swap(&arr[pos-1],&arr[pos-2]);
else
swap(&arr[pos-1],&arr[pos]);
for(int i=0;i<num;i++)
{
if(arr[i] > mobile)
{
if(d[arr[i]-
1]==0)
d[arr[i]-1] = 1;
else
```



```
#include <stdio.h>
#include <stdlib.h>
int flag = 0; int
swap(int *a,int *b)
{
int t =*a;
*a = *b;
*b = t;
}
int search(int arr[],int num,int mobile)
{
int g;
for(g=0;g<num;g++)
{
if(arr[g] == mobile)
return g+1;
else {
flag++;
}
}
return -1;
}
int find_Moblie(int arr[],int d[],int num)
{
int mobile = 0;
int mobile_p =0;
int i;
for(i=0;i<num;i++)
{
if((d[arr[i]-1] == 0) && i != 0)
{
if(arr[i]>arr[i-1] && arr[i]>mobile_p)
{
mobile = arr[i];
mobile_p = mobile;
}
else
{
flag++;
}
```



```
Class Solution {
public:
static bool cmp(const string lhs,const string rhs){
if(lhs.length()==rhs.length()){
int i=0;
while(i<lhs.length() && lhs[i]==rhs[i]){
i++;
}
return lhs[i]<rhs[i];
}
return lhs.length()<rhs.length();
}
string kthLargestNumber(vector& nums, int k) {
sort(nums.begin(),nums.end(),cmp);
int n=nums.size();
return nums[n-k];
}
};
```

ved

Ln 12, Col 2

nums =

```
["0","0"]
```

k =

```
2
```