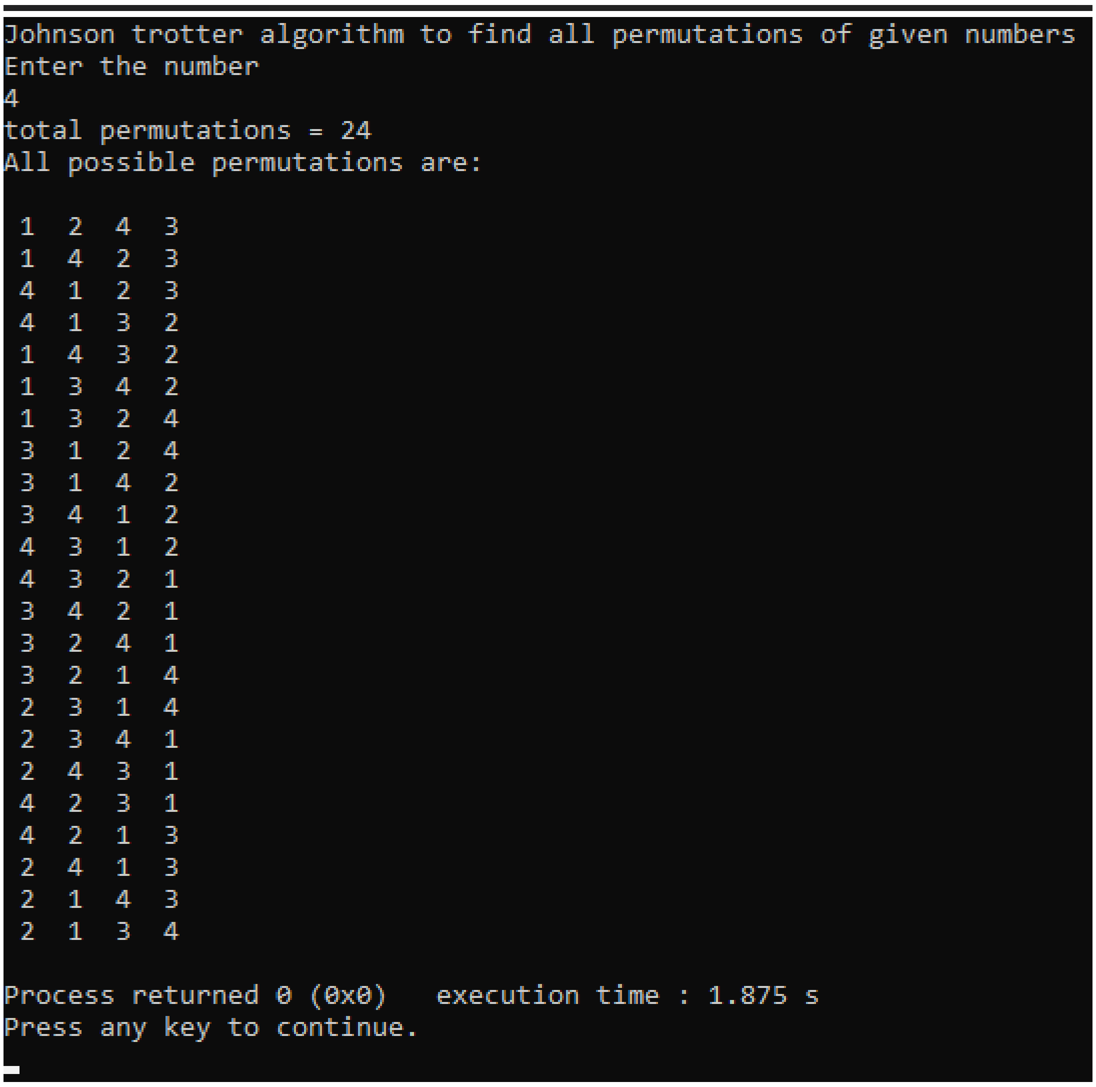
# Johnson Trotter Algorithm

```
#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_Mobile(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++;
            }
        } 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_Mobile(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
                d[arr[i] - 1] = 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;
}
```

## OUTPUT


```
Johnson trotter algorithm to find all permutations of given numbers
Enter the number
4
total permutations = 24
All possible permutations are:

 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

Process returned 0 (0x0)   execution time : 1.875 s
Press any key to continue.
```

## **Find the Kth Largest Integer in the Array-LEETCODE**

You are given an array of strings nums and an integer k. Each string in nums represents an integer without leading zeros.

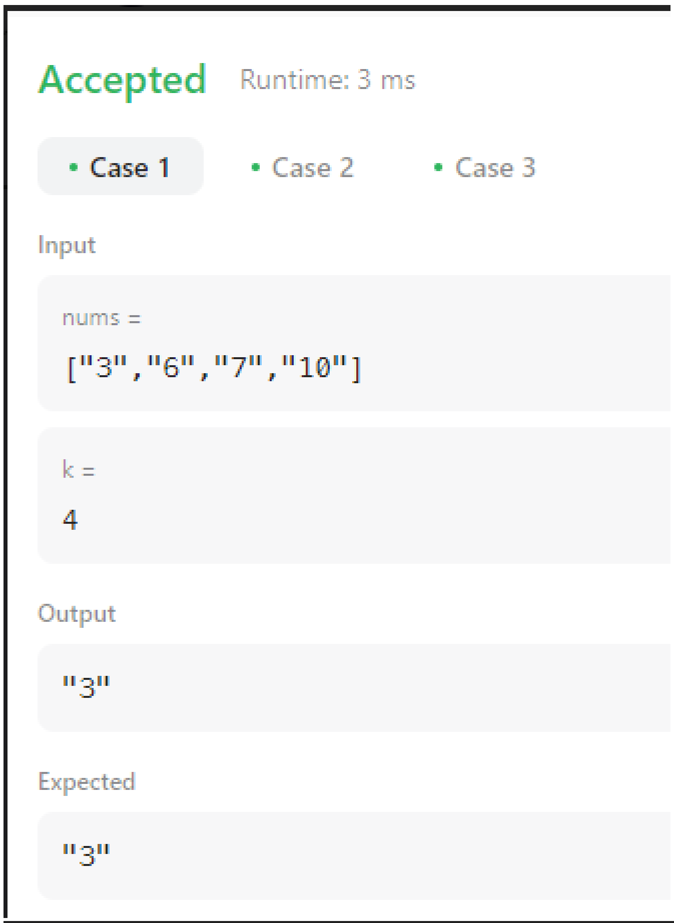
Return *the string that represents the* k$^{th}$ *largest integer in* nums.

Note: Duplicate numbers should be counted distinctly. For example, if nums is ["1","2","2"], "2" is the first largest integer, "2" is the second-largest integer, and "1" is the third-largest integer.

**CODE**

```
int compare(const void *a, const void *b) {
    const char *str1 = *(const char **)a;
    const char *str2 = *(const char **)b;

    int len1 = strlen(str1);
    int len2 = strlen(str2);
    if (len1 != len2) {
        return len2 - len1;
    }
    return strcmp(str2, str1);
}

char* kthLargestNumber(char **nums, int numsSize, int k) {
    qsort(nums, numsSize, sizeof(char*), compare);
    return nums[k - 1];
}
```

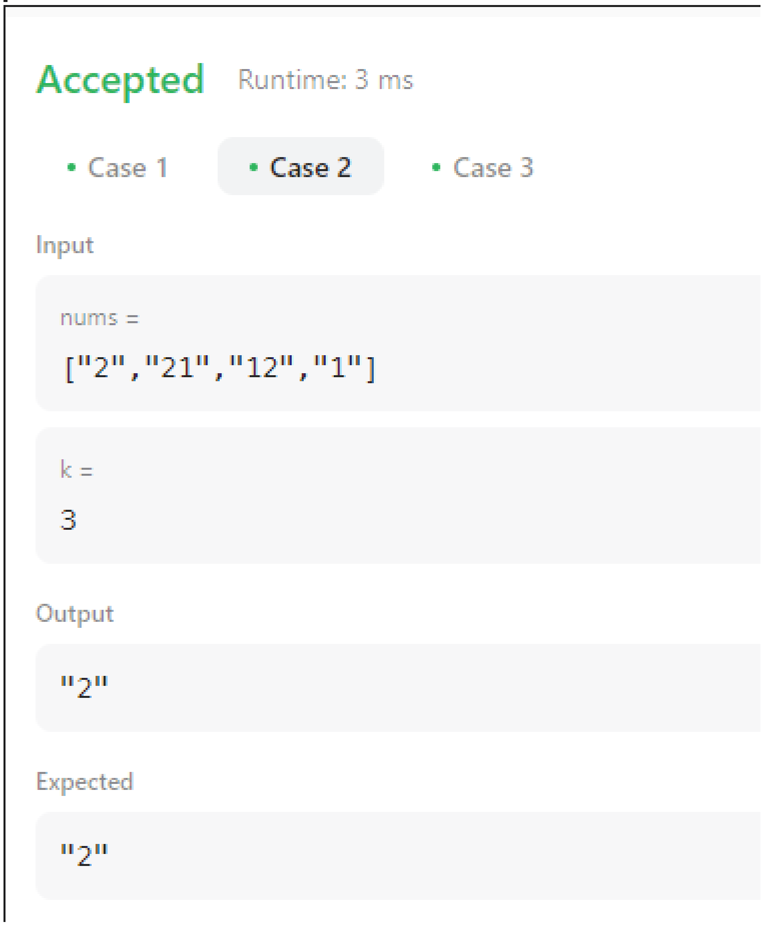
**OUTPUT**


Accepted Runtime: 3 ms

• Case 1 • Case 2 • Case 3

Input

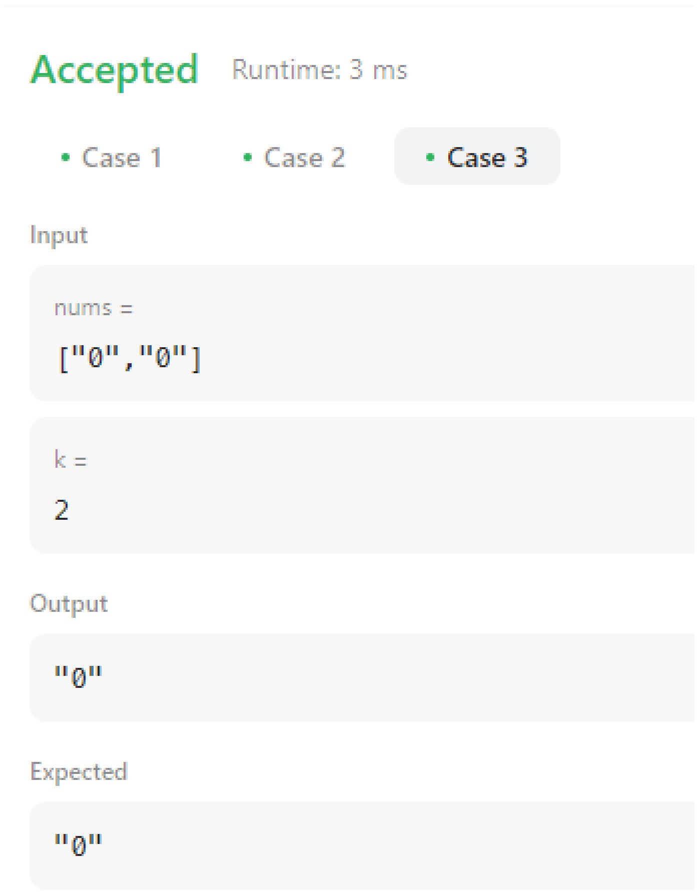
nums =
```
["3","6","7","10"]
```
k =
```
4
```
Output
```
"3"
```
Expected
```
"3"
```


Accepted Runtime: 3 ms

• Case 1 • Case 2 • Case 3

Input

nums =
```
["2","21","12","1"]
```
k =
```
3
```
Output
```
"2"
```
Expected
```
"2"
```



Accepted Runtime: 3 ms

• Case 1 • Case 2 • Case 3

Input

nums =

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

k =

```
2
```

Output

```
"0"
```

Expected

```
"0"
```