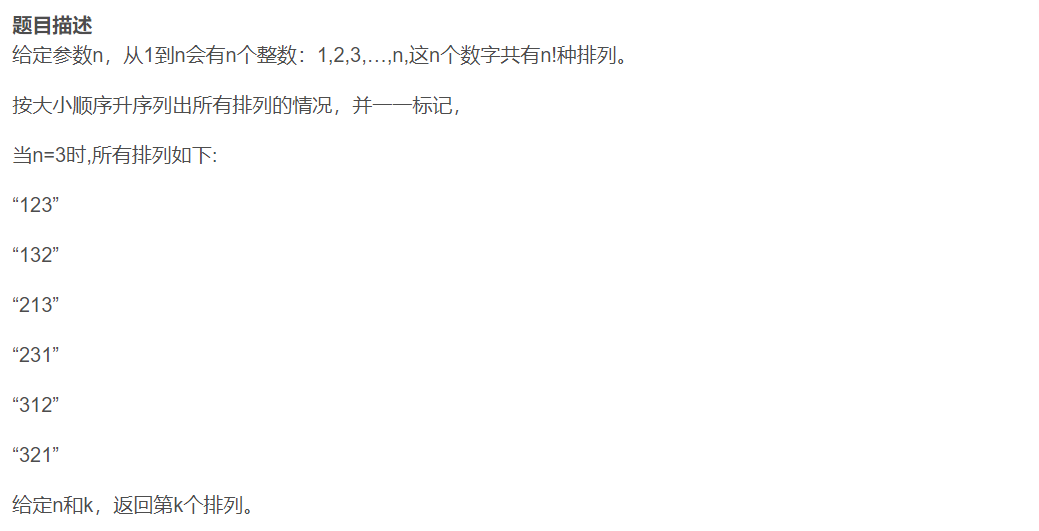
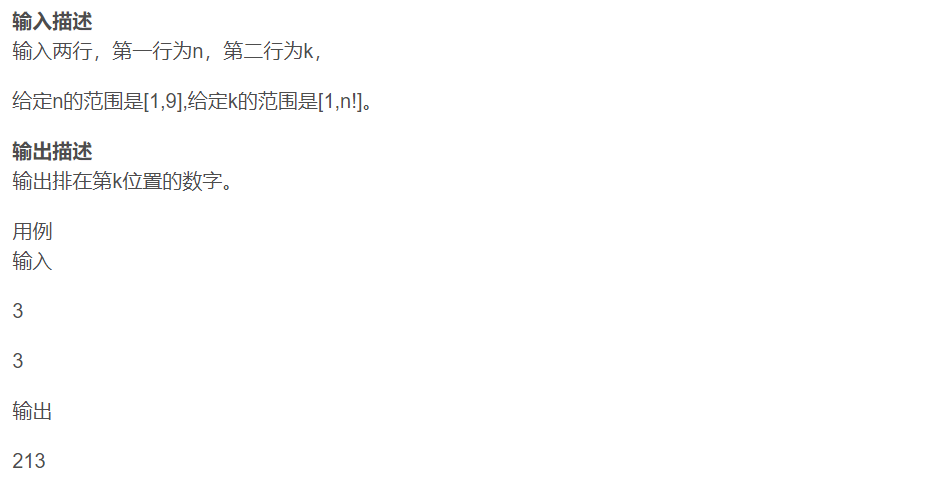
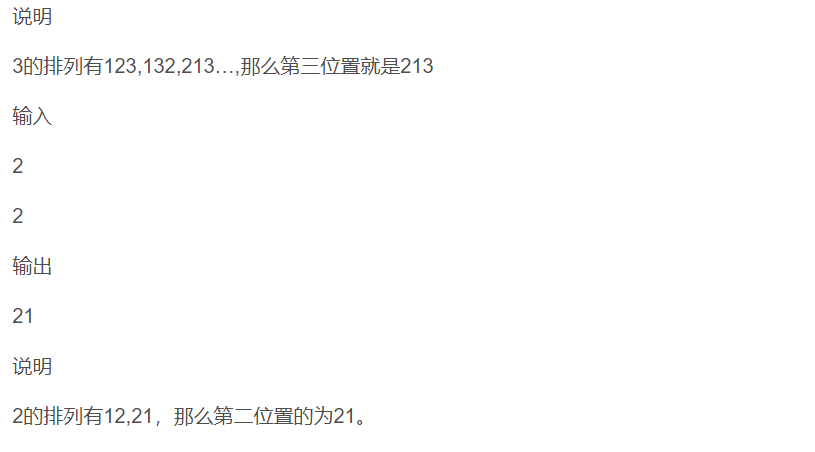
# **E卷-第k个排列[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 读取n和k

int n = scanner.nextInt();

int k = scanner.nextInt();

List<Integer> list = new ArrayList<>();

for (int i = 1; i <= n; i++) {

list.add(i); // 生成1到n的序列

}

List<List<Integer>> permutations = new ArrayList<>();

// 生成所有排列

generatePermutations(list, 0, permutations);

// 打印第k-1个排列

for (int num : permutations.get(k - 1)) {

System.out.print(num);

}

System.out.println();

}

private static void generatePermutations(List<Integer> list, int start,

List<List<Integer>> result) {

if (start == list.size() - 1) {

result.add(new ArrayList<>(list));

return;

}

for (int i = start; i < list.size(); i++) {

Collections.swap(list, start, i);

generatePermutations(list, start + 1, result);

Collections.swap(list, start, i);

}

}

}



import itertools

n, k = map(int, input().split())

res = list(itertools.permutations(x for x in range(1, n+1)))

print("".join(map(str, res[k-1])))



#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric> // std::iota

using namespace std;

int main() {

int n, k;

cin >> n >> k; // 读取n和k

vector<int> vec(n);

iota(vec.begin(), vec.end(), 1); // 生成1到n的序列

vector<vector<int>> permutations;

// 生成所有排列

do {

permutations.push\_back(vec);

} while (next\_permutation(vec.begin(), vec.end()));

// 打印第k-1个排列

for (auto num : permutations[k-1]) {

cout << num;

}

cout << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void reverse(int \*arr, int start, int end) {

while (start < end) {

swap(&arr[start], &arr[end]);

start++;

end--;

}

}

int next\_permutation(int \*arr, int n) {

int i = n - 2;

while (i >= 0 && arr[i] >= arr[i + 1]) {

i--;

}

if (i < 0) return 0;

int j = n - 1;

while (arr[j] <= arr[i]) {

j--;

}

swap(&arr[i], &arr[j]);

reverse(arr, i + 1, n - 1);

return 1;

}

int main() {

int n, k;

scanf("%d %d", &n, &k); // 读取n和k

int \*arr = malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

arr[i] = i + 1; // 生成1到n的序列

}

int count = 0;

do {

count++;

if (count == k) {

for (int i = 0; i < n; i++) {

printf("%d", arr[i]);

}

printf("\n");

break;

}

} while (next\_permutation(arr, n));

free(arr);

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.question("", (answer) => {

let [n, k] = answer.split(" ").map(Number); // 读取n和k

let arr = Array.from({ length: n }, (\_, i) => i + 1); // 生成1到n的序列

// 生成所有排列

let perms = permutations(arr);

// 打印第k-1个排列

console.log(perms[k - 1].join(""));

rl.close();

});

function permutations(arr) {

let result = [];

if (arr.length === 0) return [[]];

for (let i = 0; i < arr.length; i++) {

let current = arr.slice(i, i + 1);

let remaining = arr.slice(0, i).concat(arr.slice(i + 1));

let innerPerms = permutations(remaining);

for (let perm of innerPerms) {

result.push(current.concat(perm));

}

}

return result;

}



package main

import (

"fmt"

)

func main() {

var n, k int

fmt.Scan(&n, &k) // 读取n和k

arr := make([]int, n)

for i := 0; i < n; i++ {

arr[i] = i + 1 // 生成1到n的序列

}

permutations := permute(arr)

// 打印第k-1个排列

for \_, num := range permutations[k-1] {

fmt.Print(num)

}

fmt.Println()

}

func permute(nums []int) [][]int {

var result [][]int

var backtrack func(first int)

n := len(nums)

backtrack = func(first int) {

if first == n {

perm := make([]int, n)

copy(perm, nums)

result = append(result, perm)

}

for i := first; i < n; i++ {

nums[first], nums[i] = nums[i], nums[first]

backtrack(first + 1)

nums[first], nums[i] = nums[i], nums[first]

}

}

backtrack(0)

return result

}