1. **设有上三角矩阵（aij)n×n ，将其上三角元素逐行存于数组B(1:m)中，使得B[k]= aij，且k= f1(i)+ f2(j)+c。试推导函数 f1、f2 和常数项 c，其中1≤i，j≤n。**

**第i行元素个数为f(i) = n - i + 1。**

**1 ~i - 1行元素个数为： [n + n - (i - 1) + 1] \* (i - 1) / 2 = -0.5 \* i ^ 2 + (n + 1.5) \*i - n - 1；**

**第i行的aij之前元素个数为：j - i - 1；**

**因此，k = -0.5 \* i ^ 2 + (n + 0.5) \* i + j - n - 1。**

**推出f1(i) = -0.5 \* i ^ 2 + (n + 0.5) \* i**

**f2(j) = j**

**c = -n - 1**

1. **设计一个算法，将数组 A(1:n) 中的元素循环右移 k 位，要求只用一个元素的附加空间，元素移动或交换次数为O(n)。**

#include <iostream>

using namespace std;

#define ARRAY\_LEN 10

void moveArray(int \*a, int k)

{

k %= ARRAY\_LEN;

int gcd = 1; // 最大公约数

for (int i = 1; i <= k; i++)

{

if (ARRAY\_LEN % i == 0 && k % i == 0)

gcd = i;

}

int temp;

for (int j = 0; j < gcd; j++)

{

int i = j;

do

{

i %= ARRAY\_LEN;

temp = a[(i + k) % ARRAY\_LEN], a[(i + k) % ARRAY\_LEN] = a[j], a[j] = temp;

i += k;

} while (i != ARRAY\_LEN + j);

}

return;

}

int main()

{

int k, a[ARRAY\_LEN] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

cout << "原数组：";

for (auto i : a)

cout << i << ' ';

cout << endl << "请输入向右移动的位数：";

cin >> k;

moveArray(a, k);

for (auto i : a)

cout << i << ' ';

return 0;

}

**运行该程序，输出结果如下：**

**原数组：1 2 3 4 5 6 7 8 9 10**

**请输入向右移动的位数：2**

**9 10 1 2 3 4 5 6 7 8**

1. **求下列广义表运算的结果**
2. **Head[((a,b),(c,d))] = (a,b)**
3. **Tail[((a,b),(c,d))] = ((c,d))**
4. **Head[Tail[((a,b),(c,d))]] = (c,d)**
5. **Tail[Head[((a,b),(c,d))]] = (b)**
6. **Head[Tail[Head [((a,b),(c,d))]]] = b**
7. **Tail[Head[Tail [((a,b),(c,d))]]] = (d)**
8. **利用Head和Tail运算将下列广义表中的原子c取出来**
9. **L1=(a,b, c,d) = Head[Tail[Tail[L1]]]**
10. **L2=((a,b),(c,d)) =Head [Head[Tail[L2]]]**
11. **L3=(((a),(b), (c),(d))) = Head[Head[Tail[Tail[Head[L3]]]]]**
12. **L4=(a,(b),((c)),(((d)))) = Head[Head[Head[Tail[Tail[L4]]]]]**
13. **L5=((((a))),((b)),(c),d) = Head[Head[Tail[Tail[L5]]]]**
14. **L6=((((a),b),c),d) = Head[Tail[Head[L6]]]**
15. **L7=(a,(b,(c,(d)))) = Head[Tail[Tail[L7]]]**
16. **L8=(a,(b,(c),d)) = Head[Head[Tail[Head[Tail[L8]]]]]**