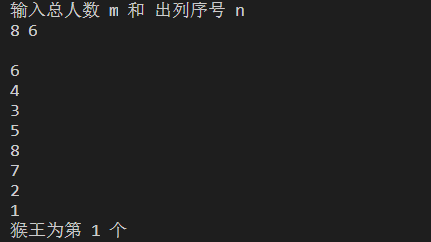
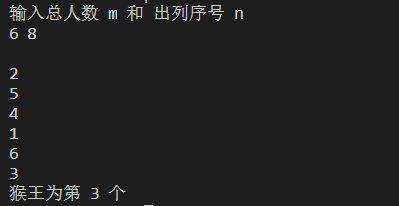
1. 顺序储存

测试用例：i) 总人数 m = 8，出列序号 n = 6

ii) m = 6, n = 8

运行结果：





具体算法：1）将数组的下标表示 m 个人的顺序，i 来记录数的次序，数组元素的值设置为0表示还未出列，end 表示结束循环的数组元素，start 是开始的数组元素，total 是出列的人数。

2）当数到出列序号，将对应数组的值设置为1，表示出列，同时记录出列人的序号。若出列的是每次循环开始的位置，start 的值加 1，使下一次循环在数组上的开始往后移动一位；若出列的是每次循环结束的位置，end 的值减 1，使下一次的循环结束提前一个。

3）当所有人全部出列（total == m）结束计数。打印出列的人的次序。

#include <stdio.h>

int main(){

    int n, m;

    printf("输入总人数 m 和 出列序号 n \n");

    scanf("%d%d", &m, &n);

    int end, start = 0;

    end = m - 1;

    int a[m] = {0}, b[m] = {0};

    int total = 0, i, j;

    int q;

    for(i = 0, j = 1; total != m;){

        if(a[i] != 1){

            if(j == n){

                a[i] = 1;

                b[total] = i + 1;

                total++;

                j = 0;

                q = i;

                if(i == start){   //当移去的是开始一次循环的人

                    while(a[q] == 1){

                        start++;

                        q++;

                    }

                }

                if(i == end){   //当移去的是结束一次循环的人

                    while (a[q] == 1){

                        q--;

                        end--;

                    }

                }

            }

            i++;

            j++;

            if(i > end){

                i = start;

            }

        }

        else{

            i++;

        }

    }

    for(i = 0; i < total; i++){

        printf("\n%d", b[i]);

}

printf("\n猴王为第 %d 个", b[total - 1]);

    return 0;

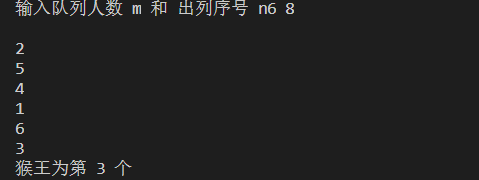
}

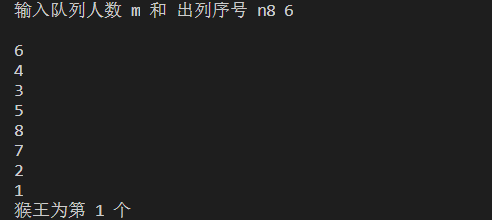
2、链表储存

测试用例：i) 总人数 m = 8，出列序号 n = 6

ii) m = 6, n = 8

运行结果：





具体算法：1）建立动态链表，同时对每个结点记录其次序。

2）把最后一个结点连接到head的下一个结点，释放head。

3）利用i计数，当数到出列序号，删除这一结点，并释放空间。

4）当所有人全部出列，结束计数。

5）打印出列的人的序号。

#include <stdio.h>

#include <stdlib.h>

#define LEN sizeof(struct list)

typedef struct list{

    int num;

    struct list \*next;

}List;

int main(){

    int m, n, total = 0, i = 1;

    printf("输入队列人数 m 和 出列序号 n");

    scanf("%d%d", &m, &n);

    int a[m];

    list \*head, \*p1, \*p2;

    head = p1 = (List\*)malloc(LEN);

    head->next = NULL;

    if(NULL == p1)

    {

        printf("内存分配失败，程序终止\n");

        exit(-1);

    }

    do{

        p2 = (List\*)malloc(LEN);

        p2->num = i;

        p1->next = p2;

        p1 = p2;

        i++;

        if(i > m){

            p1->next = head->next;   //head 结点没有被计数，跳过head 结点连接到下一个结点

            break;

        }

    }while(1);

    p1 = head;

    p2 = p1->next;

    free(head);

    i = 0;

    do{

        i++;

        if(i == n){

            a[total] = p2->num;

            total++;

            i = 0;

            p1->next = p2->next; //删除结点，同时释放空间

            free(p2);

            p2 = p1;

        }

        p1 = p2;

        p2 = p2->next;

    } while(total < m);

    for(i = 0; i < m; i++){

        printf("\n%d", a[i]);

}

printf("\n猴王为第 %d 个", a[total - 1]);

    return 0;

}