猴子选大王不同实现方法介绍

猴子选大王

在m只猴子聚在一起选大王, 商定规则如下: 大家围成一圈, 按顺时针从1编号, 第一次从编号为1的开始报数, 以后循环进行, 当报到n时退出圈子, 下一只则重新从1开始报数, 圈子中剩下的最后一只猴子则为大王.

有多组测试数据. 输入的第一行是整数T(1<=T<=100), 表示随后测试数据的组数. 每组测试数据占一行, 由正整数m和n组成, 两数之间有一个空格. 2 <= m,n <= 200.

对应每组测试数据, 输出选出的大王的猴子编号.

样例输入	样例输出
4 3 2 1 23 20 1 20 3	3 1 20 20

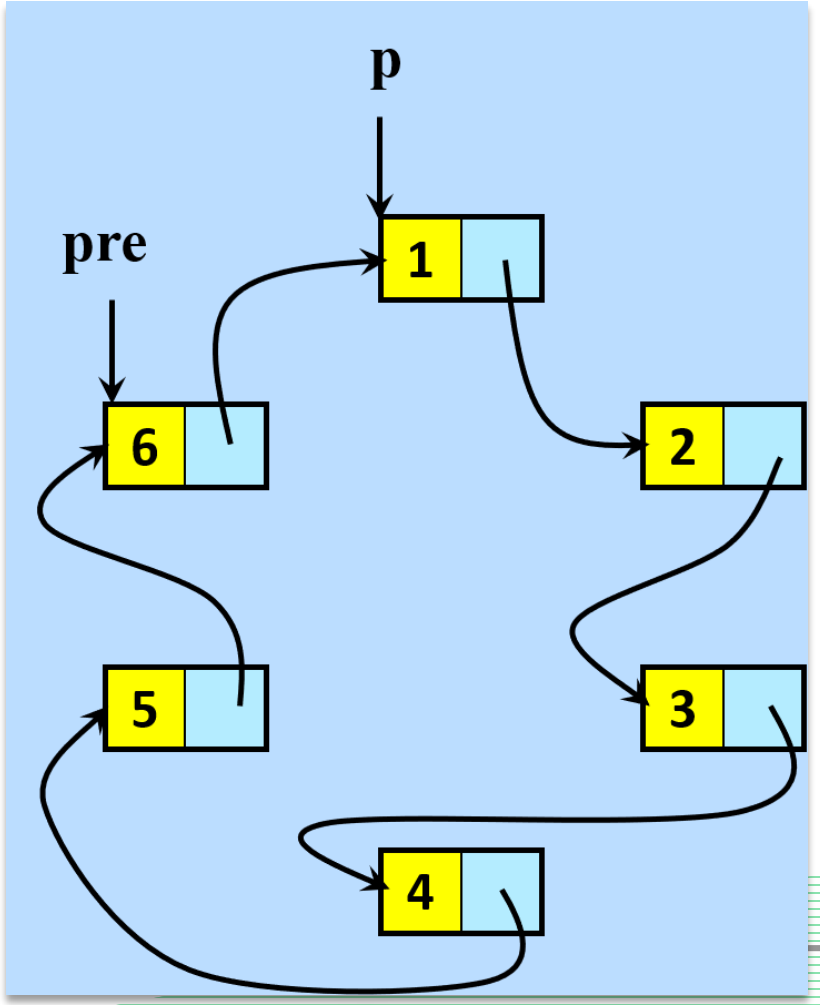
法一：利用一维数组模拟

- 用数组元素下标表示猴子编号(为方便编程，编号从0开始)，元素值用0表示该猴子在圈内，为1表示已退出圈子；
- 为实现循环，当下标达到m时，变为0（可以用%运算）；
- 当前猴子报数后，注意跳过标志为1的元素。

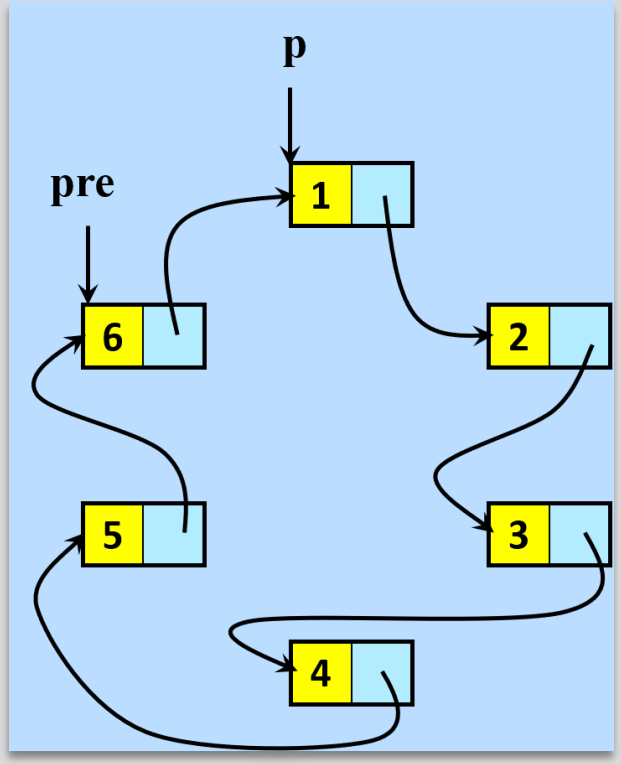
```
#include <stdio.h>
#define N 202
 int main() {
    int T;  scanf("%d", &T);
    while(T--)  {
       int i, j = 0, n, m, cnt, a[N] = {0};
       scanf("%d%d", &m, &n);
       for(i = 1; i < m; i++) {
          cnt = 0;
          while(cnt < n)  {
             while(a[j]) j = (j+1) % m;
                cnt++;
             }
             a[j] = 1;
          }
          for(i = 0; i < m; i++)
            if(!a[i])
               printf("%d\n", i+1);
    }
    return 0;
}
```

法二：利用循环链表实现（动态分配内存）

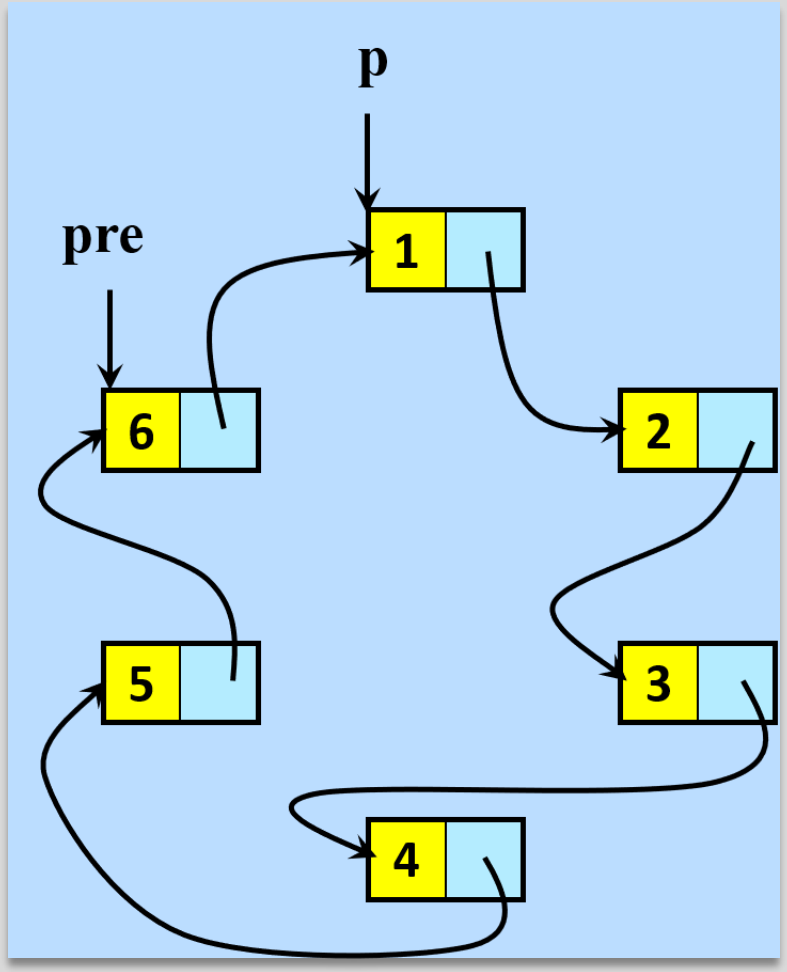
- 使用单向链表，最后一个结点的next指针指向第一个结点，形成循环链表；
- 由题目的特点，不必使用单独的头结点。



```
#include <stdio.h>
#include <stdlib.h>
typedef struct _monkey {
    int id;
    struct _monkey* next;
} monkey;
monkey* create(int m) {
    int i;
    monkey* p = (monkey*)malloc(sizeof(monkey)), *last = p;
    last->id = 1;
    for(i = 2; i <= m; i++) {
        last->next = (monkey*)malloc(sizeof(monkey));
        last = last->next;
        last->id = i;
    }
    last->next = p;
    return p;
}
```

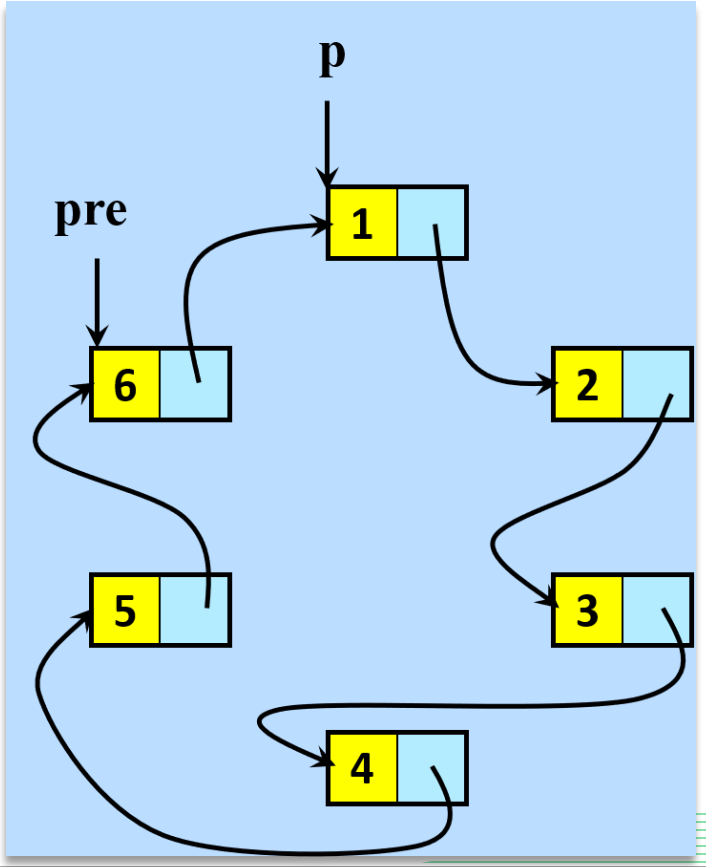


```
int main() {
    int T;  scanf("%d", &T);
    while(T--) {
        int i, n, m, cnt;
        monkey* p, *pre;
        scanf("%d%d", &m, &n);     pre = p = create(m);
        for(i = 1; i < m; i++)  {
            cnt = 1;
            while(cnt < n)  {
                pre = p;  p = p->next;  cnt++;
            }
            pre->next = p->next;
            free(p);
            p = pre->next;
        }
        printf("%d\n", p->id);     free(p);
    }
    return 0;
}
```



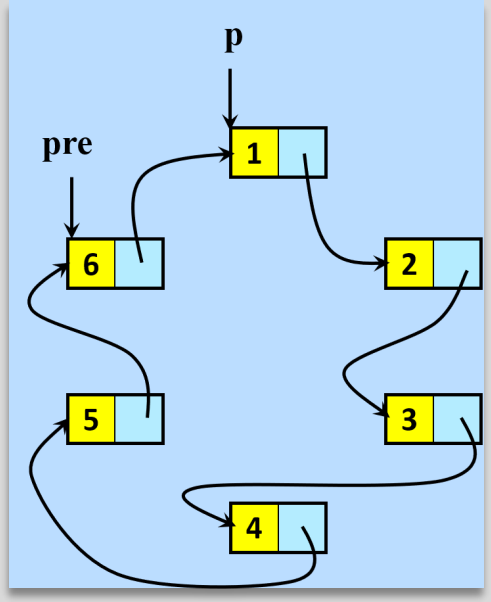
法三：利用循环链表实现（使用数组和指针）

- 使用单向链表，最后一个结点的next指针指向第一个结点，形成循环链表，注意，没有动态分配内存，所以不用free；
- 由题目的特点，不必使用单独的头结点。



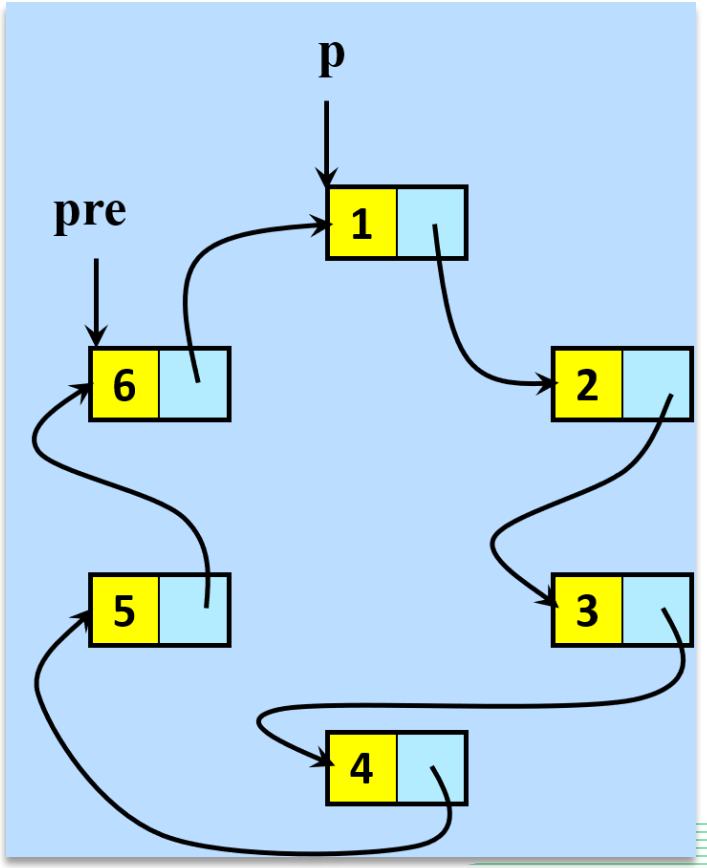
```c
#include <stdio.h>
#include <string.h>
#define N 202
typedef struct _monkey {
    int id;
    struct _monkey* next;
} monkey;
monkey mky[N];
monkey* create(int m) {
    int i;
    for(i = 0; i < m-1; i++) {
        mky[i].id = i+1;
        mky[i].next = &mky[i+1];
    }
    mky[m-1].id = m;
    mky[m-1].next = &mky[0];
    return &mky[0];
}
```

```c
int main() {
    int T;   scanf("%d", &T);
    while(T--)  {
        int i, n, m, cnt;
        monkey* p, *pre;
        scanf("%d%d", &m, &n);
        pre = p = create(m);
        for(i = 1; i < m; i++) {
            cnt = 1;
            while(cnt < n)  {
                pre = p;   p = p->next;
                cnt++;
            }
            pre->next = p->next;   p = pre->next;
        }
        printf("%d\n", p->id);
    }
    return 0;
}
```



法四：利用循环链表实现（使用数组和下标）

- 使用单向链表，最后一个结点的next指针指向第一个结点，形成循环链表，注意：这儿指针是数组下标；
- 由题目的特点，不必使用单独的头结点。

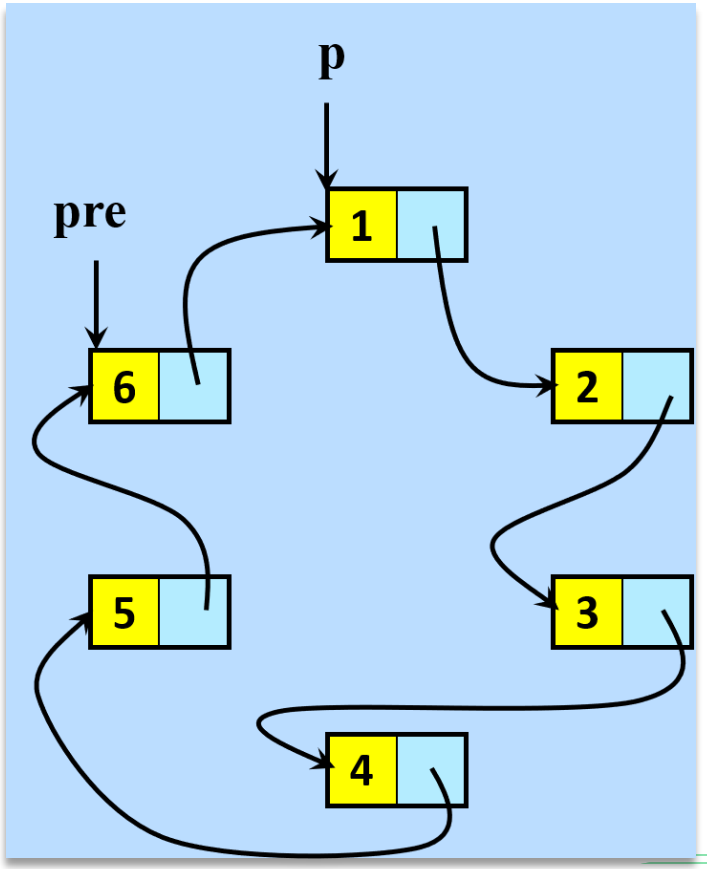


```
#include <stdio.h>
#include <string.h>
#define N 202
typedef struct _monkey {
    int id;
    int next;
} monkey;
monkey mky[N];
int create(int m) {
    int i;
    for(i = 0; i < m-1; i++)  {
        mky[i].id = i+1;
        mky[i].next = i+1;
    }
    mky[m-1].id = m;
    mky[m-1].next = 0;
    return 0;
}
```

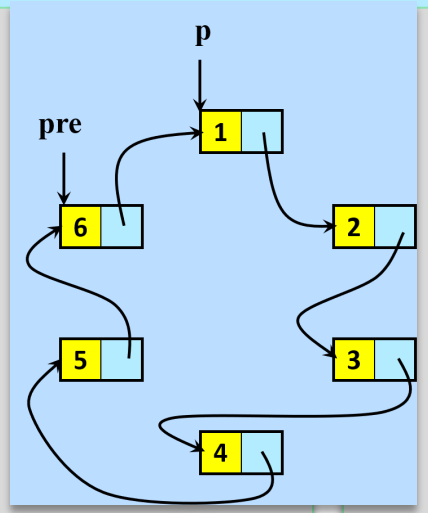
```
int main()  {
    int T;   scanf("%d", &T);
    while(T--)  {
        int i, n, m, cnt, p, pre;
        scanf("%d%d", &m, &n);
        pre = p = create(m);
        for(i = 1; i < m; i++)   {
            cnt = 1;
            while(cnt < n)  {
                pre = p;    p = mky[p].next;
                cnt++;
            }
            mky[pre].next = mky[p].next;
            p = mky[pre].next;
        }
        printf("%d\n", mky[p].id);
    }
    return 0;
}
```

法五：利用一维数组静态链表实现

- 下标表示猴子编号（从0开始），元素值记录下一只猴子的下标；
- 本方法本质上就是法四，只不过写法更简洁。



```
#include <stdio.h>
#include <string.h>
#define N 202
int mky[N];
int create(int m)
{
    int i;
    for(i = 0; i < m-1; i++)
        mky[i] = i+1;
    mky[m-1] = 0;
    return 0;
}
 int main()
 {
     int T;
     scanf("%d", &T);
     while(T--)
     {
         int i, n, m, cnt, p, pre;
```




```
    scanf("%d%d", &m, &n);
    pre = p = create(m);
    for(i = 1; i < m; i++)
    {
        cnt = 1;
        while(cnt < n)
        {
            pre = p;
            p = mky[p];
            cnt++;
        }
        mky[pre]= mky[p];
        p = mky[pre];
    }
    printf("%d\n", p+1);
  }
  return 0;
}
```