题目：约瑟夫环

班级：19184115

姓名：

学号：19063140

日期：2020.11.04

### 需求分析：

利用单向循环列表存储结构模拟约瑟夫环过程，按照出列的顺序印出个人的编号。

### 概要设计：

struct List 循环链表

void ListInit(LinkList \*l) 链表初始化

void ListClear(LinkList l) 链表清空

LinkList ListFind(LinkList l, ElemType e) 在链表中找到元素e所在位置并返回该地址

void ListInsert(LinkList l, ElemType pre, ElemType cur) 在链表元素pre后插入原素cur，若无pre元素，则重建一个只有cur元素的链表

void ListErase(LinkList l, ElemType e) 在链表中删除元素e

void Joseph(int n, int k) 模拟约瑟夫环,报到k的人退出

### 调试分析：

1. 问题
   1. 问题：写好了第一次运行，程序非正常退出

原因：访问了空指针

解决：传参时使用指针

1. 时空分析

时间复杂度为 O(n\*k+n\*n)

每次插入删除复杂度O(n) 构造链表以及删除结点总复杂度O(n\*n)

模拟约瑟夫环过程O(nk)

空间复杂度为 O(n)

### 用户使用说明：

输入两个非负整数n,k表示总人数和报k的人退出

### 测试结果：

输入

3 1

输出

第1轮退出的人的编号是1

第2轮退出的人的编号是2

第3轮退出的人的编号是3

输入

5 2

输出

第1轮退出的人的编号是2

第2轮退出的人的编号是4

第3轮退出的人的编号是1

第4轮退出的人的编号是5

第5轮退出的人的编号是3

输入

7 3

输出

第1轮退出的人的编号是3

第2轮退出的人的编号是6

第3轮退出的人的编号是2

第4轮退出的人的编号是7

第5轮退出的人的编号是5

第6轮退出的人的编号是1

第7轮退出的人的编号是4

### 附录：

1. #include <stdio.h>
2. #include <stdlib.h>
4. **typedef** **int** ElemType;
6. **typedef** **struct** List {
7. ElemType e;
8. **struct** List \*pre, \*nex;
9. } List, \*LinkList;
11. **void** ListInit(LinkList \*l) {
12. \*l = (LinkList)malloc(**sizeof**(List));
13. **if** (!l) exit(0);
14. (\*l)->e = -1;
15. (\*l)->pre = (\*l)->nex = \*l;
16. }
18. **void** ListClear(LinkList \*l) {
19. **if** (!l) **return**;
20. LinkList p = \*l, t;
21. **do** {
22. t = p->nex;
23. free(p);
24. p = t;
25. } **while** (p != \*l);
26. }
28. LinkList ListFind(LinkList l, ElemType e) {
29. **if** (!l) **return** NULL;
30. LinkList p = l;
31. **do** {
32. **if** (p->e == e) **return** p;
33. p = p->nex;
34. } **while** (p != l);
35. **return** NULL;
36. }
38. // 在 pre 前插入 cur
39. **void** ListInsert(LinkList \*l, ElemType pre, ElemType cur) {
40. LinkList p = ListFind(\*l, pre);
41. **if** (!p) {
42. ListClear(l);
43. ListInit(l);
44. (\*l)->e = cur;
45. **return**;
46. }
47. LinkList u = (LinkList)malloc(**sizeof**(List));
48. **if** (!u) exit(0);
49. u->e = cur;
50. u->pre = p;
51. u->nex = p->nex;
52. p->nex->pre = u;
53. p->nex = u;
54. }
56. **void** ListErase(LinkList l, ElemType e) {
57. LinkList p = ListFind(l, e);
58. **if** (!p) **return**;
59. p->pre->nex = p->nex;
60. p->nex->pre = p->pre;
61. free(p);
62. }
64. **void** Joseph(**int** n, **int** k) {
65. LinkList l;
66. ListInit(&l);
67. **for** (**int** i = 1; i <= n; ++i) ListInsert(&l, i-1, i);
69. **for** (**int** t = 1; t <= n; ++t) {
70. **for** (**int** i = 1; i < k; ++i) l = l->nex;
71. printf("第%d轮退出的人的编号是%d\n", t, l->e);
72. l = l->nex;
73. ListErase(l, l->pre->e);
74. }
75. }
77. **int** main() {
78. **int** n, k;
79. printf("请输入总人数和退出的人所报数字：");
80. scanf("%d %d", &n, &k);
81. Joseph(n, k);
82. **return** 0;
83. }