第七次练习赛题解

A

难度	考点
1	模拟

问题分析

根据第一行获取的进制数n，将第二行输入按字符读入并模拟n进制数的转换即可。

参考代码

```
#include <stdio.h>
#include <ctype.h>

int main() {

    int n;
    char c;
    long long ans = 0;
    scanf("%d", &n);
    while ((c = getchar()) != EOF) {
        if (isdigit(c)) {
            ans = ans * n + c - '0';
        }
    }
    printf("%lld\n", ans);

    return 0;
}
```

B

难度	考点
1	代码基础

问题分析

考虑数据量较小，我们可以在输入后对数组遍历两次：第一次得到最大的总分，第二次依次输出总分和最大总分相等的学生信息。

参考代码

```c
#include <stdio.h>

typedef struct node{//创建一个名为 'struct node' 的结构体
    char name[15];//定义成员
    int chinese;
    int math;
    int eng;
    int score;
}stuinf;

int main(){
    stuinf stu[10005];
    int i, j, max = 0;
    for(i = 0; scanf("%s", stu[i].name) != EOF; i++){
        scanf("%d%d%d", &stu[i].chinese, &stu[i].math, &stu[i].eng);
        /* 直接调用结构体内部的成员时，我们使用 '.' 访问.
        而通过指针调用结构体成员时，我们则需要使用 '->' 访问.
        因此，该 scanf 函数可以等价于下面的写法：
        scanf("%d%d%d", &(stu + i)->chinese, &(stu + i)->math, &(stu + i)->eng);
        在该写法中，我们使用了 '->' 访问指针 stu+i 指向的结构体元素成员 */
        stu[i].score = stu[i].chinese + stu[i].math + stu[i].eng;
        if(max < stu[i].score)
            max = stu[i].score;
    }
    for(j = 0; j < i; j++)
        if(stu[j].score == max)
            printf("%s %d %d %d\n", stu[j].name, stu[j].chinese, stu[j].math,
stu[j].eng);
    return 0;
}
```

C

难度	考点
2	递归

问题分析

本题的关键在于发现待求解问题的递归性质。对于给定的长为n的序列，我们可以将其平均分成两个长度为$\frac{n}{2}$的子序列（n为偶数），或者长度为$\frac{n}{2}$和长度为$\frac{n}{2}+1$的子序列，我们将包含n的子序列中所有的数减去$\frac{n}{2}$，得到的新数列是之前包含1的子集，之后我们只需要考虑长为$\frac{n}{2}$的问题的答案就行，递推关系如下$f(n)=f(\frac{n}{2})+1$且$f(1)=1$

```
#include<stdio.h>

int n;

int f(int n){
    if(n == 1)  return 1;
    return 1+f(n/2);
}

int main(){
    scanf("%d", &n);

    printf("%d", f(n));
    return 0;
}
```

D

难度	考点
2	循环控制

问题分析

这道问题的关键在于如何打印每一行前面的空格数，我们以高为16的树为例。

树最宽的地方是在第二部分的最后一行，长度为$\frac{h}{2}\times 2-1=h-1=15$。

所以每一行前面的空格数最多为$\frac{(h-1)-1}{2}=\frac{h}{2}-1=7$。

对于前两个部分，起始的第一行前都为7个空格，随后每行前空格依次减1。

对于第三个部分，设$d=(h/16)\times 2+1=3$，前面的空格长度恒定为$\frac{h}{2}-1-\frac{d-1}{2}=\frac{h-(d+1)}{2}=6$。

```
       * //前面7个空格
      *** // 前面6个空格
     *****
    *******
       * //前面7个空格
      *** //前面6个空格
```

```
     ***** //前面5个空格
    *******
   *********
  ***********
 *************
*************** //长度为15
      | | //前面6个空格
      | |
      | |
      |_|
```

参考代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define ll long long
int main()
{
    int h;
    scanf("%d",&h);;
    for(int i=1;i<=h/4;i++) //打印第一部分
    {
        for(int j=1;j<=h/2-i;j++) putchar(' '); //前面的空格长度为h/2-i
        for(int j=1;j<=2*i-1;j++) putchar('*'); //*的长度为2*i-1
        putchar('\n');
    }
    for(int i=1;i<=h/2;i++) //打印第二部分，与第一部分类似
    {
        for(int j=1;j<=h/2-i;j++) putchar(' ');
        for(int j=1;j<=2*i-1;j++) putchar('*');
        putchar('\n');
    }
    int d=(h/16)*2+1;
    for(int i=1;i<=h/4;i++) //打印第三部分
    {
        for(int j=1;j<=h/2-(d+1)/2;j++) putchar(' '); //前面的空格长度为h/2-(d+1)/2
        putchar('|');
        if(i!=h/4) for(int j=1;j<=d-2;j++) putchar(' '); //不是最后一行，中间为空格
        if(i==h/4) for(int j=1;j<=d-2;j++) putchar('_'); //最后一行，中间为_
        putchar('|');
        putchar('\n');
    }
    // system("pause");
    return 0;
}
```

E

难度	考点
2	快速排序

问题分析

本题主要考查多条件的结构体快速排序，对于字符串的字典序排序其实可以直接调用strcmp函数，并不需要一个个字符判断。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct stu {
    char name[20];
    char id[10];//也可以用int类型读入学号，输出的时候使用%08d即可
    double sco;
} stu;

int cmp(const void *p1, const void *p2) {
    stu a = *(stu *) p1;
    stu b = *(stu *) p2;
    if (a.sco < b.sco)
        return 1;
    else if (a.sco > b.sco)
        return -1;
    else if (strcmp(a.id, b.id) < 0)
        return -1;
    else if (strcmp(a.id, b.id) > 0)
        return 1;
    else if (strcmp(a.name, b.name) > 0)
        return 1;
    else
        return -1;
}

int n;
stu stus[10000];

int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%s %s %lf", stus[i].name, stus[i].id, &stus[i].sco);
    }
    qsort(stus, n, sizeof(stu), cmp);
    for (int i = 0; i < n; i++) {
```

```
        printf("%s %s %.4f\n", stus[i].name, stus[i].id, stus[i].sco);
    }
}
```

F

难度	考点
3	递归，搜索

问题分析

本题为图的连通块问题。

我们可以从任意一个水坑的地方开始搜索，每到达一个水坑位置就把它变为旱地，直到与它八连通的所有水坑位置都变为旱地。

再重复上面的过程，直到所有格子都变为旱地为止，部分解析见代码。

参考代码

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
void dfs(int x,int y);
int N,M;
char field[110][110];
int main()
{
    scanf("%d%d",&N,&M);//输入field的大小
    for(int v=0;v<N;v++)
    {
        scanf("%s",field[v]);
    }
    int  res=0;
    for(int i=0;i<N;i++)
    {
        for(int j=0;j<M;j++)
        {
            if(field[i][j]=='W')
            {

                dfs(i,j);//遇见一个水坑，就去找它周围八个方向有没有水池
                res++;//当搜索完成时，代表无联通，答案加1
            }
        }
```

```
    }
    printf("%d",res);
    return 0;
}

void dfs(int x,int y)
{
    field[x][y]='*';//搜过的位置直接赋值为*,旱地。
    for(int dx=-1;dx<=1;dx++)//分别代表八个方向
    {
        for(int dy=-1;dy<=1;dy++)
        {
            int nx=x+dx;int ny=y+dy;//注意判断边界条件，不要超出园子的范围
            if(nx>=0&&nx<N&&ny>=0&&ny<M&&field[nx][ny]=='W')//如果在给定的field中找到另一个
水池，再对这个水池周围继续进行dfs
            {
                dfs(nx,ny);
            }
        }
    }
    return;//尽可能完成全部的搜索之后再返回，不要提前返回

}
```

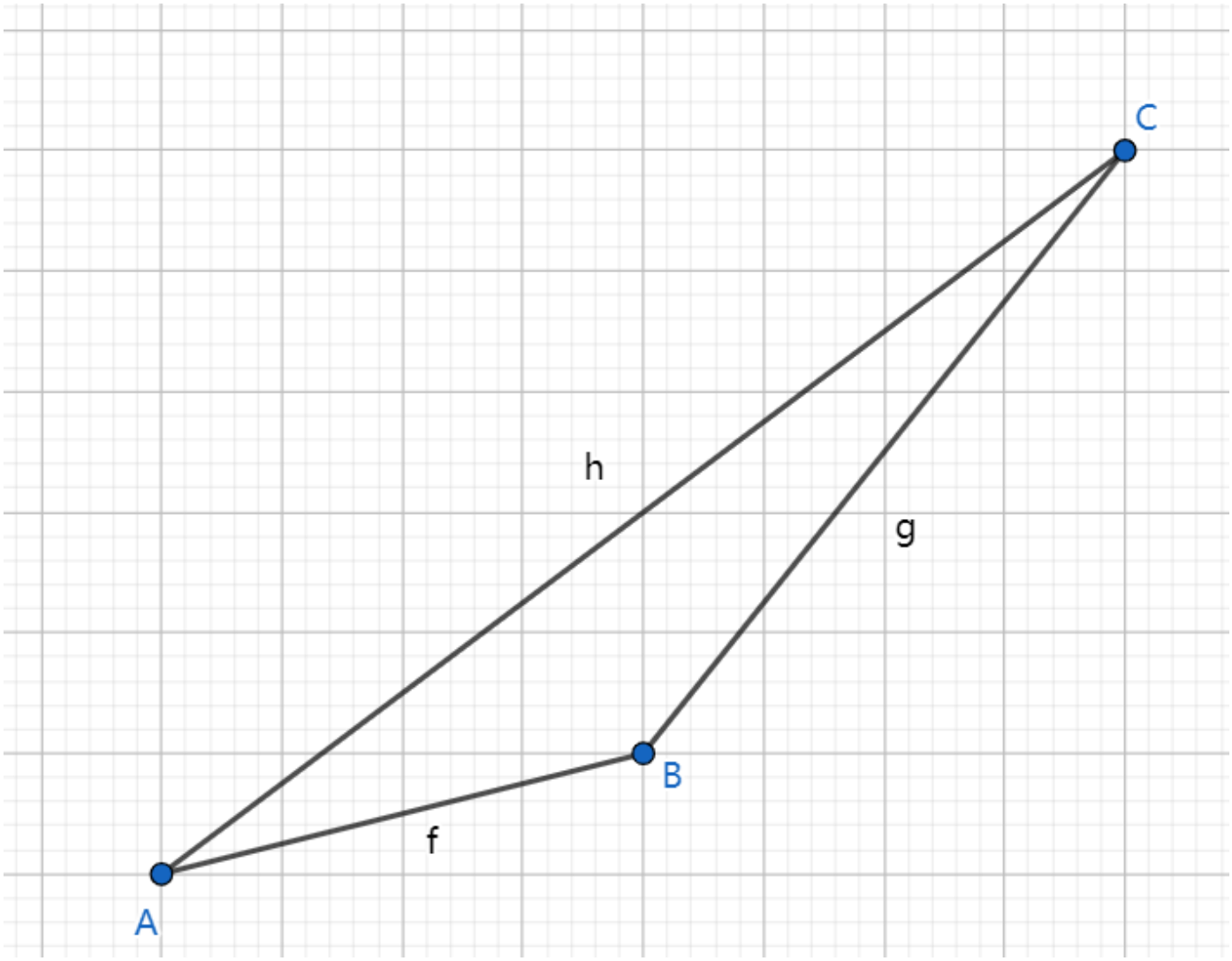
G

难度	考点
3	快排，结构体

问题分析

朴素的做法是枚举所有的点对并计算斜率，求其中的最大值。这种方法的时间复杂度是$O(n^2)$及需要进行n^2次左右的计算，在本题的数据范围中，最多需要1×10^{10}次，而一般情况下$1s$内所能进行的运算次数在$10^7 - 10^8$左右，显然会TLE，那我们能不能在$O(nlogn)$的复杂度情况下解决这个问题呢?

我们考虑最大斜率产生的条件，如下图所示，任取三个点，我们会得到三条可能存在最大斜率的直线，显然，h的斜率是介于f和g之间的，所以最大斜率只可能在横坐标相邻的两点之间产生（题目中保证了横坐标都不相同），所以我们只需将所有点按横坐标排序，依次计算相邻两点的斜率，从中选取最大值即可，采用*qsort*的话时间复杂度就可以降为$nlogn$。



参考代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define ll long long
typedef struct Node
{
    int x,y;
}Node; //定义结构体Node
Node node[500010];
int cmp(const void *a,const void *b) //定义以横坐标排序的cmp
{
    Node A=(*(Node *)a),B=(*(Node *)b);
    if(A.x<B.x) return -1;
    if(A.x>B.x) return 1;
    return 0;
}
int main()
```

```
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d%d",&node[i].x,&node[i].y);
    qsort(node+1,n,sizeof(Node),cmp); //因为数组下标是从第1位开始存的，所以qsort中node+1
    double k=-0x7fffffff;
    //0x7fffffff=2^31-1=2147483647是16进制表示方法，此处也可以设置成理论最小斜率-10^7
    for(int i=1;i<n;i++)
    {
        int cx=node[i+1].x-node[i].x;
        int cy=node[i+1].y-node[i].y;
        double t=(double)cy/(double)cx; //计算相邻两点的斜率
        if(t>k) k=t; //更新最大斜率
    }
    printf("%.3lf",k);
    return 0;
}
```

H 多项式相乘

难度	考点
4	链表

本题参考思路已经在hint中给出，使用链表并借助两个指针来实现多项式每项两两相乘。两个指针中有一个指针固定，另一个指针进行遍历，遍历完后再移动固定的指针，直到两个指针都遍历到多项式末尾，插入元素时注意保持结果的有序性。和C4的多项式相加一样，本题的多项式指数可能非常大，因此不能使用下标来存储指数数据。
本题也有同学使用数组，最后将结果进行排序，这种方法的正确性是显然的。但链表相比数组，插入和删除有更高的效率，并无需最后对链表进行排序。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node *Poly;
struct Node
{
    long long coe; //系数
    long long exp;    //指数
    Poly next;
};
Poly p,ps,p1,p2,ps1,ps2,t,end,q;
int n,m;//项数
long long x,y;
int main(){
    scanf("%d %d",&n,&m);
    p1 = (Poly)malloc(sizeof(struct Node));
```

```
    p2 = (Poly)malloc(sizeof(struct Node));
    p1->next = NULL;
    p2->next = NULL;
    ps=p1;
    Poly newPoly;
    //输入
    for (int i = 0; i < n; i++)
    {
        scanf("%lld %lld", &x, &y);
        newPoly = (Poly)malloc(sizeof(struct Node));
        newPoly->coe = x;
        newPoly->exp = y;
        newPoly->next = NULL;
        ps->next = newPoly;
        ps=ps->next;
    }
    ps=p2;
    for (int i = 0; i < m; i++)
    {
        scanf("%lld %lld", &x, &y);
        newPoly = (Poly)malloc(sizeof(struct Node));
        newPoly->coe = x;
        newPoly->exp = y;
        newPoly->next = NULL;
        ps->next = newPoly;
        ps=ps->next;
    }
    p = (Poly)malloc(sizeof(struct Node));
    p->next = NULL;
    ps1=p1->next;
    ps2=p2->next;
    end=p;
    int flag=1;//判断是否第一个结点
    while (ps1 != NULL)
    {

        ps2=p2->next;
        while (ps2 != NULL)
        {
            long long c, e;
            c = ps1->coe * ps2->coe;
            e = ps1->exp + ps2->exp;
            if (flag)
            {
                flag = 0;
                t = (Poly)malloc(sizeof(struct Node));
                t->coe = c;
                t->exp = e;
                t->next = NULL;
```

```
                end->next = t;
                end = end->next;
            }
            else
            {
                q = p;
                //遍历出插入位置
                while (q->next != NULL&&q->next->exp > e)
                    q = q->next;
                if (q->next!=NULL&&q->next->exp == e)    //合并
                {
                    if (q->next->coe+c==0)  //删掉系数为0的项
                    {
                        q->next = q->next->next;
                    }
                    else{
                        q->next->coe=q->next->coe+c;
                    }
                }
                else
                {
                    t = (Poly)malloc(sizeof(struct Node));
                    t->coe = c;
                    t->exp = e;
                    t->next = q->next;
                    q->next = t;
                }
            }

            ps2=ps2->next;
        }
        ps1=ps1->next;
    }
    //输出
    ps=p->next;
    while (ps != NULL)
    {
        printf("%lld %lld ", ps->coe, ps->exp);
        ps=ps->next;
    }
}
```

I 结构体大练习

难度	考点
4	结构体、结构体指针

题目解析

本题按照题目要求完成每步操作即可，注意插入操作时若移动结构体内容会因操作数过多导致TLE，因此本题提供两种做法，示例代码 1 为使用结构体指针的做法，实例代码 2 为移动结构体数组下标的做法。

示例代码 1

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>

typedef struct Stu{
    char name[55];
    int id;
    char gender[10];
    long long phone;
    struct Stu *next;//
}stu,*stulink;//定义结构体和结构体的指针类型

int main()
{
    int n,m,i,j,a,id;
    long long phone_new;
    scanf("%d %d",&n,&m);

    stulink now=(stulink)malloc(sizeof(stu)),newstu,endstu;//创建初始空间
    stulink start=now;//定义链表开始节点

    scanf("%s %d %s %lld",now->name, &now->id, now->gender, &now->phone);//读入初始变量
    now->next=NULL;//注意未使用的指针应当在定义时指向NULL

    for(i=1;i<n;i++)
    {
        newstu=(stulink)malloc(sizeof(stu));//申请结构体空间
        newstu->next=NULL;
        now->next=newstu;//在链表末尾添加节点
        now=newstu;
        scanf("%s %d %s %lld",now->name, &now->id, now->gender, &now->phone);
    }
    endstu=now;

    for(i=0;i<m;i++)
    {
        scanf("%d",&a);
        if(a==1)//在链表末尾添加节点
        {
            newstu=(stulink)malloc(sizeof(stu));
```

```
            newstu->next=NULL;
            now=start;
            while(now->next!=NULL)//查找链表末尾节点
            {
                now=now->next;
            }
            now->next=newstu;
            scanf("%s %d %s %lld",newstu->name, &newstu->id, newstu->gender, &newstu-
>phone);
            n++;
        }
        else if(a==2)//修改联系方式
        {
            scanf("%d %lld",&id,&phone_new);
            now=start;
            while(now!=NULL&&now->id!=id)//查找指定节点
            {
                now=now->next;
            }
            now->phone=phone_new;
        }
        else if(a==3)//删除指定节点
        {
            scanf("%d",&id);
            if(start->id==id)//如果需要删除的是开始节点，需要处理start指向的节点
            {
                start=start->next;
                n--;
                continue;
            }
            now=start;
            while(now->next!=NULL&&now->next->id!=id)//找到指定节点的前一节点
            {
                now=now->next;
            }
            now->next=now->next->next;//将前一节点的链接跳过需要删除的节点，实现删除目的
            n--;
        }
        else if(a==4)//插入节点
        {
            newstu=(stulink)malloc(sizeof(stu));
            newstu->next=NULL;
            scanf("%s %d %s %lld",newstu->name, &newstu->id, newstu->gender, &newstu-
>phone);
            scanf("%d",&id);
            now=start;
            while(now!=NULL&&now->id!=id)//找到指定节点
            {
                now=now->next;
```

```
            }
            newstu->next=now->next;//将新生成的节点指向指定节点的下一节点，指定节点指向新生成的节点，实现插入目的
            now->next=newstu;
            n++;
        }
    }

    now=start;
    for(i=0;i<n;i++)//按顺序输出
    {
        printf("%s %d %s %lld\n",now->name, now->id, now->gender, now->phone);
        now=now->next;
    }
    return 0;
}
```

示例代码 2

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>

typedef struct Stu{
    char name[55];
    int id;
    char gender[10];
    long long phone;
}stu;//定义结构体

int main()
{
    stu info[15055];//定义结构体数组
    int n,m,i,j,k,a,id,in[15055];//in存储结构体数组下标
    long long phone_new;
    scanf("%d %d",&n,&m);

    for(i=0;i<15055;i++) in[i]=-1;//将不使用的下标置为-1
    for(i=0;i<n;i++)
    {
        scanf("%s %d %s %lld",info[i].name, &info[i].id, info[i].gender,&info[i].phone);//保存初始数据和下标
        in[i]=i;
    }

    for(i=0;i<m;i++)
```

```
    {
        scanf("%d",&a);
        if(a==1)//在末尾插入节点
        {
            scanf("%s %d %s %lld",info[n].name, &info[n].id, info[n].gender
,&info[n].phone);
            in[n]=n;
            n++;
        }
        else if(a==2)//修改手机号
        {
            scanf("%d %lld",&id,&phone_new);
            for(j=0;j<n;j++)
            {
                if(info[j].id==id)
                {
                    info[j].phone=phone_new;
                    break;
                }
            }
        }
        else if(a==3)//删除节点
        {
            scanf("%d",&id);
            for(j=0;j<n;j++)
            {
                if(in[j]!=-1&&info[in[j]].id==id)//查找节点时需要保证下标不是-1的条件在前，避
免数组越界
                {
                    in[j]=-1;//将需要删除的节点下标改为-1，标记为不使用的节点
                    break;
                }
            }
        }
        else if(a==4)//在指定节点后插入节点
        {
            scanf("%s %d %s %lld",info[n].name, &info[n].id, info[n].gender
,&info[n].phone);
            scanf("%d",&id);
            for(j=0;j<n;j++)//查找指定节点
            {
                if(in[j]!=-1&&info[in[j]].id==id) break;
            }
            for(k=n;k>j+1;k--)
            {
                in[k]=in[k-1];//将指定节点后的所有下标向后移动一位
            }
            in[j+1]=n;//插入下标
            n++;
```

```
        }
    }

    for(i=0;i<15055;i++)//按顺序输出所有下标不是-1的节点
    {
        if(in[i]!=-1) printf("%s %d %s %lld\n",info[in[i]].name, info[in[i]].id,
info[in[i]].gender ,info[in[i]].phone);
    }
    return 0;
}
```

J 来种树吧

难度	考点
5	二叉树

题目分析

本题是典型的“二叉树模型”，其中前序遍历，中序遍历，后序遍历在以二叉树为基础的数据结构中十分常用。

二叉树模型在C语言中的结构如下：

```
struct treeNode {
    int value;
    struct treeNode *left;
    struct treeNode *right;
};
```

关于如何将输入表示成二叉树，有两种方式：

1. 采取“队列”的数据结构，每当新增一个节点，将节点加入队尾，并连接在队首的左节点/右节点上，当队首节点的左右均有连接时，将其移出队列，从而第二个节点成为队首，直到全部输入结束。
2. 通过观察发现，节点`i`一定连接在节点`i/2`上。

关于如何正确输出二叉树，采用的方式是递归，递归的详细步骤在题目中已经表示的较为明确，结合示例代码即可。

示例代码1（队列）

```
#include <stdio.h>

struct treeNode {
    int value;
    struct treeNode *left;
    struct treeNode *right;
};
```

```c
struct treeNode queue[1 << 15];
void pre_order(struct treeNode* node);
void mid_order(struct treeNode* node);
void post_order(struct treeNode* node);


int main()
{
    int n = 0, i = 0, j = 0;
    scanf("%d", &n);
    scanf("%d", &queue[0].value);

    for (i = 1; i < (1 << n) - 1; i++) {
        scanf("%d", &queue[i].value);
        if (queue[j].left == NULL) {
            queue[j].left = &queue[i];
        }
        else if (queue[j].right == NULL) {
            queue[j].right = &queue[i];
            j++;
        }
    }

    struct treeNode* base = &queue[0];
    pre_order(base);
    printf("\n");
    mid_order(base);
    printf("\n");
    post_order(base);
    printf("\n");
    return 0;
}

void pre_order(struct treeNode* node)
{
    printf("%d ", node->value);
    if (node->left != NULL) {
        pre_order(node->left);
    }
    if (node->right != NULL) {
        pre_order(node->right);
    }
    return;
}

void mid_order(struct treeNode* node)
{
    if (node->left != NULL) {
        mid_order(node->left);
```

```
    }
    printf("%d ", node->value);
    if (node->right != NULL) {
        mid_order(node->right);
    }
    return;
}

void post_order(struct treeNode* node)
{
    if (node->left != NULL) {
        post_order(node->left);
    }
    if (node->right != NULL) {
        post_order(node->right);
    }
    printf("%d ", node->value);
    return;
}
```

示例代码2（找规律）

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<math.h>

typedef struct IN{
    int num;
    int left;
    int right;
}In;
In in[1111];

void front(int i)
{
    printf("%d ",in[i].num);
    if(in[i].left!=-1) front(in[i].left);
    if(in[i].right!=-1) front(in[i].right);
    return;
}

void mid(int i)
{
    if(in[i].left!=-1) mid(in[i].left);
    printf("%d ",in[i].num);
```

```
    if(in[i].right!=-1) mid(in[i].right);
    return;
}

void back(int i)
{
    if(in[i].left!=-1) back(in[i].left);
    if(in[i].right!=-1) back(in[i].right);
    printf("%d ",in[i].num);
    return;
}

int main()
{
    int n,i,j;
    scanf("%d",&n);
    n=pow(2,n)-1;
    for(i=1;i<=n;i++)
    {
        scanf("%d",&in[i].num);
        in[i].left=-1;
        in[i].right=-1;
        j=i/2;
        if(j*2==i) in[j].left=i;
        else in[j].right=i;
    }
    front(1);
    printf("\n");
    mid(1);
    printf("\n");
    back(1);
    return 0;
}
```

K 一定要有一位巫妖王

难度	考点
循环链表	6

题目分析

本题的模型为循环遍历的模型，可以采用的方式为构建一个链表并首尾相接，使链表能够如题目所述一般构成一个圈，从而能实现对数据的循环遍历以及删除操作，之后的逻辑过程较为清晰。

示例代码

```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    char name[35];
    struct Node* next;
};

int main()
{
    int n = 0, m = 0, k = 0;
    scanf("%d%d%d", &n, &m, &k);

    struct Node* head, * tail, * pre;
    head = (struct Node*)malloc(sizeof(struct Node));
    head->next = NULL;
    tail = head;
    for (int i = 0; i < n; i++) {
        scanf("%s", tail->name);
        pre = tail;
        tail = (struct Node*)malloc(sizeof(struct Node));
        tail->next = NULL;
        pre->next = tail;
    }
    pre->next = head;
    free(tail);

    for (int i = 1; i < m; i++) {
        pre = pre->next;
    }
    tail = pre;

    while (pre->next != pre) {
        for (int i = 0; i < k; i++) {
            pre = tail;
            tail = tail->next;
        }
        pre->next = tail->next;
        free(tail);
        tail = pre;
    }

    puts(pre->name);
    free(pre);
    return 0;
}
```

扩展

本题的原型是“约瑟夫环”模型，关于该模型，在数学上的求解方式为构造一个递推式，并求得封闭表达式。关于此模型，有兴趣的同学可以自行研究，此处不予叙述。