Problem A. 助教机器人

难度	考点
1	字符串匹配

题目分析

使用`strcmp`函数对比输入数据与给定的问题，然后输出对应答案。

示例代码

```
#include<stdio.h>
#include<string.h>
char question[200];
int main(){
    while(gets(question)!=NULL){
        if (strcmp(question,"助教，我的代码样例是对的，为什么交上去WA了？")==0)
            puts("本地对了就是对了，交上去WA说明评测机有问题。");
        else if (strcmp(question,"助教，我的代码为什么过不去样例啊。")==0)
            puts("调试调试。\n");
        else if (strcmp(question,"助教,我的代码REG是怎么回事？")==0)
            puts("SIGESGV大约是数组越界了,SIGFPE大约是除以零了。");
        else if (strcmp(question,"助教我的代码OE了。")==0)
            puts("大约是数组越界了。");
        else if (strcmp(question,"助教我的代码CE了。")==0)
            puts("百度一下报错信息。");
        else if (strcmp(question,"助教我的代码什么都不输出。")==0)
            puts("可能是死循环了，如果显示process exited with return value 3221225477
等非零数，可以百度百度这个数字的携带的错误信息。");
        else puts("PARDON?");
    }
}
```

Problem B. 回文魔咒

难度	考点
2	字符串、回文串的判断

题目分析

按题目要求循环判断字符串**去除数字**且忽略大小写后的是否是回文即可。

示例代码

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

```
#include<string.h>
#define MN 1000+5
char c[MN];
int n;
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i){
        scanf("%s",c);
        int len=strlen(c);
        int m=0;
        for(int j=0;j<len;++j)
            if('0'<=c[j]&&c[j]<='9');
            else
                c[m++]=c[j];
        for(int j=0;j<m;++j)
            if('A'<=c[j]&&c[j]<='Z')
                c[j]=c[j]-'A'+'a';
        int flag=1;
        for(int j=0;j<m;++j)
            if(c[j]!=c[m-j-1])
                flag=0;
        if(flag)printf("P1@lin&rOrne\n");
        else printf("$tr1ng\n");
    }
    return 0;
}
```

Problem C. xf 做统计

难度	考点
2	递归

题目分析

使用题目中的递归式可得$C_n=\frac{4n-2}{n+2}C_{n-1}$，然后递归求解即可

示例代码

```
#include<stdio.h>
int t;
long long Catalan(int n)
{
    if(n==1||n==0)
        return 1;
    return Catalan(n-1)*(4*n-2)/(n+1);
}
int main()
{
    scanf("%d",&t);
    printf("%lld",Catalan(t));
    return 0;
}
```

Problem D. 数位拆解

难度	考点
2	循环、数组

题目分析

按题目要求依次枚举 $[l,r]$ 中的数，并把它们依次数位拆解即可。

示例代码

```
#include <stdio.h>
int main()
{
    int l, r;
    int cnt[10] = {0};
    scanf("%d %d", &l, &r);
    for (int i = l; i <= r; i++)
    {
        int tmp = i;
        do
        {
            cnt[tmp % 10]++;
            tmp /= 10;
        } while (tmp);
    }
    for (int i = 0; i <= 9; i++)
        printf("%d ", cnt[i]);
    return 0;
}
```

Problem E. 神奇矩阵的阶乘

难度	考点
3	二维数组，循环

题目分析

本题考查二维数组的简单应用。

本题解给出一种十分朴素的解法，但是此解法并不适用于n太大的情况。思路具体看代码注释。

想更快运行出程序可以试着搜索快速幂算法自行学习。

示例代码

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

```c
int m[105][105]; //存方阵m
int n[105][105]; //存m的i次方
int tmp[105][105]; //用作临时矩阵，临时存储 n 与 m 的矩阵乘积
int main()
{
    int num;
    scanf("%d",&num);
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num; j++) {
            scanf("%d", &m[i][j]);
            n[i][j] = m[i][j]; //初始化 n矩阵 等于 m矩阵
        }
    }
    for (int a = 0; a < num - 1; a++) {
        for (int i = 0; i < num; i++) {//求n 与 m 相乘的结果
            for (int j = 0; j < num; j++) {
                for (int k = 0; k < num; k++) {
                    tmp[i][j] |= n[i][k] & m[k][j];
                }
            }
        }
        for (int i = 0; i < num; i++) { //令 n矩阵 等于 tmp矩阵
            for (int j = 0; j < num; j++) {
                n[i][j] = tmp[i][j];
                tmp[i][j] = 0; //同时令tmp矩阵为零矩阵
            }
        }
    }
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num; j++) {
            printf("%d ", n[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Problem F. 寻找五子相连

难度	考点
3	二维数组的应用

题目分析

此题的思路很明确，但是计算时的细节较多。主要是是作为大家预习二维数组的尝试。不过仍有需要注意的地方：

1，题目明确了棋子可以被重复计算。

2，在枚举斜45°角的情况时，要考虑到越界的问题。

下面的做法比较“暴力”，方法是分别按照横着、竖着、斜45°的方向枚举每个格子。如果这个格子没有棋子，则跳过；如果有棋子，则看它往后四个棋子是否与它相同。

横、竖方向比较直观，难点在于斜45°的情况。具体方法以注释的形式写在了代码里。

另外，此题有更优的做法。假如寻找的是 M 子相连， N 和 M 均很大，该如何优化?

示例代码

```
#include<stdio.h>
int N, a[12][12];
int main(){
    int i, j, k, t, flag, cnt_black = 0, cnt_white = 0;
    scanf("%d", &N);
    for(i = 1; i <= N; i++)
        for(j = 1; j <= N; j++)
            scanf("%d", &a[i][j]);
    for(i = 1; i <= N; i++){ //横向寻找
        for(j = 1; j <= N - 4; j++){
            if(a[i][j] == 0) continue;
            flag = 1;
            for(k = 1; k <= 4; k++){ //固定了a[i][j]，以它为起点，横着往后枚举4个棋子看是否相同。
                if(a[i][j] != a[i][j + k]){
                    flag = 0; //若后面的4个有一个不同，则以a[i][j]开始的5个不是五子相连。
                    break;
                }
            }
            if(flag == 1){
                if(a[i][j] == 1) cnt_black++;
                else cnt_white++;
            }
        }
    }
    for(j = 1; j <= N; j++){ //竖向寻找
        for(i = 1; i <= N; i++){
            if(a[i][j] == 0) continue;
            flag = 1;
            for(k = 1; k <= 4; k++){ //固定了a[i][j]，以它为起点，竖着往后枚举4个棋子看是否相同。
                if(a[i][j] != a[i + k][j]){
                    flag = 0;
                    break;
                }
            }
            if(flag == 1){
                if(a[i][j] == 1) cnt_black++;
                else cnt_white++;
            }
        }
    }
    for(i = 1; i <= N; i++)
        for(j = 1; j <= N; j++){ //枚举a[i][j]，以a[i][j]为起点，分别向右下、左下找五子相连
            if(a[i][j] == 0) continue;
            if(i + 4 <= N && j + 4 <= N){ //满足从a[i][j]往右下后面至少还有4个棋子。
                flag = 1;
```

```
                for(k = 1; k <= 4; k++){ //往右下找。
                    if(a[i][j] != a[i + k][j + k]){
                        flag = 0;
                        break;
                    }
                }
                if(flag == 1){
                    if(a[i][j] == 1) cnt_black++;
                    else cnt_white++;
                }
            }
            if(i + 4 <= N && j - 4 >= 1){ //满足从a[i][j]往左下后面至少还有4个棋子。
                flag = 1;
                for(k = 1; k <= 4; k++){ //往左下找。
                    if(a[i][j] != a[i + k][j - k]){
                        flag = 0;
                        break;
                    }
                }
                if(flag == 1){
                    if(a[i][j] == 1) cnt_black++;
                    else cnt_white++;
                }
            }
    }
    printf("%d\n%d", cnt_black, cnt_white);
    return 0;
}
```

Problem G. 异或和的和

难度	考点
3	位运算、循环

题目分析

通过分析，可以得知，$a \oplus b \leq a + b$，因此所有数都在同一组时所有组数的**异或和的和**最小，此时答案为所有数的异或和。

示例代码

```
#include <stdio.h>
int main()
{
    int ans = 0;
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
    {
        int x;
        scanf("%d", &x);
        ans ^= x;
```

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

Problem H. 循环填数

难度	考点
4	循环、二维数组

题目分析

按题意要求循环填数即可，这里提供两个减少代码量的小技巧。

- 因为填数呈螺旋状因此每一层的正方形边长是逐渐缩小的（即第一层填 n 个数后转换方向，第二层却只填 $n-2$ 个数后就转换方向），但其实转换方向的时机是无需计算的，在填数时若当前位置超出了 $n \times n$ 正方形范围或已经被填数了就转换到下一个方向即可。
- 可以用一个方向数组：`D[4][2]={{0,1},{1,0},{0,-1},{-1,0}}`来进行位置的变化。

示例代码

```
#include<stdio.h>
#define N 1005
int a[N][N],D[4][2]={{0,1},{1,0},{0,-1},{-1,0}};
int main(){
    int n,d;
    scanf("%d%d",&n,&d);
    for(int i=1,j=1,k=0,num=0;num<d*d;num++){
        a[i][j]=num%n+1;
        i+=D[k][0],j+=D[k][1];
        if(i<1 || i>d || j<1 || j>d || a[i][j]){
            i-=D[k][0],j-=D[k][1];
            k=(k+1)%4;
            i+=D[k][0],j+=D[k][1];
        }
    }
    for(int i=1;i<=d;i++)
        for(int j=1;j<=d;j++)
            printf("%d%c",a[i][j],j==d?10:32);
}
```

Problem I. 注释消失术

难度	考点
5	字符串

题目分析

在每次遇见 `\\` 和 `\*` 的时候就开始循环向后查找注释结束的地方即可。

示例代码

```
#include<stdio.h>
#define N 1010
char buf[N];
int i = 0, flag = 0;
char mark;
int main(){
    fread(buf, 1, N, stdin);
    while(buf[i]){
        if(buf[i] == '\'' || buf[i] == '\"'){ // meet quotation mark
            if(mark == buf[i] && flag) // end of quotation
                flag = 0;
            else if(!flag){
                mark = buf[i];
                flag = 1;
            }
            putchar(buf[i]);
        }
        else if(!flag && buf[i] == '/'){
            i++;
            if(buf[i] == '*'){
                i++;
                while(1){
                    if(buf[i] == '\0'){
                        i--;
                        break;
                    }
                    if(buf[i] == '*' && buf[i+1] == '/'){
                        i++;
                        break;
                    }
                    i++;
                }
            }
            else if(buf[i] == '/'){
                i++;
                while(buf[i] != '\n' && buf[i] != '\0') i++;
                i--;
            }
            else
                putchar(buf[--i]);
        }
        else putchar(buf[i]);
        i++;
    }
    return 0;
}
```

Problem J. kk的同声传译

难度	考点
6	字符串、模拟

题目分析

要点：理解数字与英文表示的转换关系，并编写函数对转换过程进行模拟；使用变量记录当前单词种类为纯数字还是一般单词，如果为一般单词则原样输出，如果为数字则需要进行处理；由于单词间空格数量不一，不能直接使用`scanf`或`sscanf`，而应当逐字符进行处理。

示例代码

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

char ones[][17] = {"zero", "one", "two", "three", "four",
                   "five", "six", "seven", "eight", "nine"};
char tenOnes[][17] = {"", "eleven", "twelve", "thirteen", "fourteen",
                      "fifteen", "sixteen", "seventeen", "eighteen",
"nineteen"};
char tens[][17] = {"", "ten", "twenty", "thirty", "forty",
                   "fifty", "sixty", "seventy", "eighty", "ninety"};

typedef int bool;
const int true = (0 == 0), false = (0 != 0);
#define isChar(c) ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))
#define isDigit(c) (c >= '0' && c <= '9')
#define isSpace(c) (c == ' ' || c == '\n')
bool requireSpace;

void base_num2str(int num) { // 0 < num < 1000
    if (num / 100 != 0) {
        if (requireSpace) putchar(' ');
        printf("%s hundred", ones[num / 100]);
        requireSpace = true;
    }
    if (10 < num % 100 && num % 100 < 20) {
        if (requireSpace) putchar(' ');
        printf("%s", tenOnes[num % 100 - 10]);
        requireSpace = true;
    } else {
        if (num / 10 % 10 != 0) {
            if (requireSpace) putchar(' ');
            printf("%s", tens[num / 10 % 10]);
            requireSpace = true;
        }
        if (num % 10 != 0) {
            if (requireSpace) putchar(' ');
            printf("%s", ones[num % 10]);
            requireSpace = true;
        }
    }
}

void num2str(int num) {
    int one = num % 1000, thousand = num / 1000 % 1000;
```

```
    int million = num / 1000000 % 1000, billion = num / 1000000000 % 1000;
    requireSpace = false;
    if (num == 0) {
        printf("%s", ones[0]);
        return;
    }
    if (billion != 0) {
        base_num2str(billion);
        printf(" billion");
        requireSpace = true;
    }
    if (million != 0) {
        base_num2str(million);
        printf(" million");
        requireSpace = true;
    }
    if (thousand != 0) {
        base_num2str(thousand);
        printf(" thousand");
        requireSpace = true;
    }
    if (one != 0) {
        base_num2str(one);
    }
}

int main() {
    int n, i, j, k, num;
    unsigned int l;
    int status; // 0 = not in word, 1 = in num, 2 = in common word
    char iBuf[107], word[107];
    scanf("%d\n", &n);
    for (i = 0; i < n; i++) {
        fgets(iBuf, 107, stdin);
        l = strlen(iBuf);
        j = 0, k = 0, status = 0;
        while (j < l) {
            if (status == 0) {
                if (isSpace(iBuf[j])) {
                    putchar(iBuf[j]);
                } else if (isChar(iBuf[j])) {
                    putchar(iBuf[j]);
                    status = 2;
                } else if (isDigit(iBuf[j])) {
                    k = 0;
                    word[k++] = iBuf[j];
                    status = 1;
                }
            } else if (status == 1) {
                if (isSpace(iBuf[j])) {
                    word[k++] = 0;
                    sscanf(word, "%d", &num);
                    num2str(num);
                    putchar(iBuf[j]);
                    status = 0;
                } else if (isChar(iBuf[j])) {
                    word[k++] = iBuf[j];
                    word[k++] = 0;
```

```
                    printf("%s", word);
                    status = 2;
                } else if (isDigit(iBuf[j])) {
                    word[k++] = iBuf[j];
                }
            } else { // status == 2
                putchar(iBuf[j]);
                if (isSpace(iBuf[j])) {
                    status = 0;
                }
            }
            j++;
        }
        if (status == 1) {
            word[k++] = 0;
            sscanf(word, "%d", &num);
            num2str(num);
        }
    }
    return 0;
}
```