上机7解析

A找字符

难度	考点
1	二维数组的行列地址

题目分析

本题考察二维数组的行列地址概念和应用，注意要用gets()清除掉第二行的换行符，理解行地址、列地址和元素之间的转换关系。

示例代码

```
#include <stdio.h>
#include <string.h>
int main()
{
    int m, i, j, k;
    char a[25][60], enter[10];
    scanf("%d%d%d", &m, &i, &j);
    gets(enter);//清除换行符
    for(k = 0; k < m; k++)
        gets(*(a + k));
    printf("%c\n", *(*(a + i - 1) + j - 1));//输出第i行第j个字符
    printf("%s\n", *(a + i - 1) + j - 1);//输出第i行第j个字符开始的字符串
    return 0;
}
```

B班长的寄语

难度	考点
2	行、列指针

题目分析

本题考查对行列指针的简单运用，可以用算出行列指针（或数组下标）的公式来做，此时要注意行数与列数是从1开始的，与下标相差1。

示例代码

```
#include <stdio.h>
#define N 100
char stu[N+10][N+10];
int main() {
    int m,n;
    int i,j;
    int k,l;
    char tmp;
    scanf("%d%d",&m,&n);
    for(i=0; i<m; i++)
        scanf("%s",stu[i]);
    scanf("%d%d",&k,&l);

    for(j=0; j<n; j++)
        printf("%c",stu[k-1][j]);
    putchar('\n');
    int row,column;   //行 列
    if(l%n==0){
        row=l/n;
        column=n;
    }
    else{
        row=l/n+1;
        column=l%n;
    }
    for(; column<=n; column++)
    printf("%c",(*(*(stu+row-1)+column-1)));
    row++;
    for(; row<=m; row++) {
        for(column=1; column<=n; column++) {
            printf("%c",(*(*(stu+row-1)+column-1)));
        }
    }
    return 0;
}
```

C已经没什么好害怕的了

难度	考点
3	顺序结构的遍历

题目分析

这是一个中等的题，遍历一遍就行。

示例代码

```
#include<stdio.h>
int n,m;
int num1[505],num2[505];
char name[505][20];
int main()
{
    int i,j;
    scanf("%d%d",&n,&m);
    for(i=0;i<n;++i)
        scanf("%s %d",name[i],&num1[i]);
    for(i=0;i<m;++i)
        scanf("%d",&num2[i]);
    for(i=0,j=0;i<n;++i,++j)
        while(num1[i]!=num2[j])
            printf("%s\n",name[i++]);
    return 0;
}
```

D选人

难度	考点
5	生成组合数

题目分析

最直观的解法自然是深搜。唯一一个小小小小的坑点就是M个人的编程能力求和可能超出int表示范围。

示例代码

```
#include <stdio.h>

int N, M, T, flag;
int V[20], Team[20];
long long sum;

void func(int x) {
    int i;
    if (x == M) { //已选完M人
        if (sum >= T) {
```

```
            for (i = 0; i < M; i++) {
                printf("%d ", Team[i]);
            }
            printf("\n");
            flag++;
        }
    } else {
        for (i = (x > 0)? Team[x - 1] + 1: 1; i <= N - M + x + 1; i++) {
            sum += V[i - 1];
            Team[x] = i;
            func(x + 1);
            sum -= V[i - 1];
        }
    }
    return ;
}

int main() {
    int i;
    scanf("%d %d %d", &N, &M, &T);
    for (i = 0; i < N; i++) {
        scanf("%d", V + i);
    }
    func(0);
    if (!flag) {
        printf("Sorry, teacher.");
    }
    return 0;
}
```

E 又是日期转换?

难度	考点
2	字符串，循环

题目解析

签到题，注意闰年和13号就行

示例代码1

```
//此代码先按整月处理再对输入当月的天数处理
/*又是日期转换? */
#include <stdio.h>
#include <string.h>
```

```
int month_day[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
char *mon[13]={"","Jan","Feb","Mar","Apr","May","Jun",
"Jul","Aug","Sept","Oct","Nov","Dec"};

int main()
{
    char month[10];
    int day,year;
    int i,mm,ans=0,count=0;

    scanf("%s%d%d",month,&day,&year);
    if((year%4==0&&year%100!=0)||year%400==0)month_day[2]=29;

    for(i=1;i<=12;i++)
    if(strcmp(month,mon[i])==0)
    {
        mm=i;break;
    }

    for(i=12;i>mm;i--)
    ans+=month_day[i],count++;
    for(i=month_day[mm];i>day;i--)
    ans++;
    count+=(day<=13);

    printf("%d %d",ans,count);

    return 0;
}
```

示例代码2

```
//此代码全部按天数处理
/*又是日期转换? */
#include <stdio.h>
#include <string.h>
int month_day[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
char *mon[13]={"","Jan","Feb","Mar","Apr","May","Jun",
"Jul","Aug","Sept","Oct","Nov","Dec"};

int main(void)
{
    char month[10];
    int day,year;
    int i,mm,ans=0,count=0;

    scanf("%s%d%d",month,&day,&year);
    if((year%4==0&&year%100!=0)||year%400==0)month_day[2]=29;
```

```
    for(i=1;i<=12;i++)
    if(strcmp(month,mon[i])==0)
    {
        mm=i;break;
    }

    while(mm!=12||day!=31)
    {
        count+=day==13;
        ans++;
        day+=1;
        if(day>month_day[mm])day=1,mm++;
    }

    printf("%d %d",ans,count);

    return 0;
}
```

F 探险解密

难度	考点
3	输入输出

题目分析

这题本来应该出成二维数组排序的，等到把题面出好了之后发现完全可以用类似之前桶排序的思想解决问题（示例代码1），因完全算是一道简单题。每次读入第 i 条指令，并且通过格式输入分别存储操作对象、运算符、参与运算的数字（注意输入数据中间有一个空格，这里在格式输入里也加一个空格以跳过空白符，不然会把空格读进去），并把指令直接存到它该在的地方，最后从头跑一遍就可以了。

示例代码2是原计划通过二维数组的排序及字符串处理来解决问题的代码，相比起示例代码1而言就显得复杂得多。不过其中`comp()`函数的写法可以供参考（别只会傻乎乎地套ppt上的`comp()`函数，可以对它进行魔改来达到自己希望的排序效果）
P.S.如果只在`comp()`里写一句`return strcmp((char*)a,(char*)b);`可以吗?

示例代码1

```
#include <stdio.h>
int inst[200005][3];
int password[2];
int main()
{
    int N,a,i;
```

```
    scanf("%d",&N);
    for(i=0;i<N;i++)
    {
        scanf("%d ",&a);
        scanf("%c%c%d",&inst[a][0],&inst[a][1],&inst[a][2]);
    }
    for(i=1;i<=N;i++)
    {
        if(inst[i][0]=='X') a=0;
        else a=1;
        switch(inst[i][1])
        {
            case '+':password[a]+=inst[i][2];break;
            case '-':password[a]-=inst[i][2];break;
            case '*':password[a]*=inst[i][2];break;
            case '/':password[a]/=inst[i][2];break;
            case '=':password[a]=inst[i][2];break;
        }
    }
    printf("%d %d",password[0],password[1]);
    return 0;
}
```

示例代码2

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char inst[200005][25];
int password[2];
int comp(const void *a,const void *b);
int main()
{
    int N,a,i,j,obj,len,num;
    scanf("%d ",&N);
    for(i=0;i<N;i++)
    {
        fgets(inst[i],24,stdin);
        len=strlen(inst[i]);
        while(inst[i][len-1]=='\r'||inst[i][len-1]=='\n') inst[i][--len]=0;
    }
    qsort(inst,N,sizeof(inst[0]),comp);
    for(i=0;i<N;i++)
    {
        for(j=0;inst[i][j]!=' ';j++);
        obj=j+1;
        j+=3;
        for(num=0,a=1;inst[i][j]!=0;j++) num=num*10+inst[i][j]-'0';
```

```
        if(inst[i][obj]=='X') a=0;
        switch(inst[i][obj+1])
        {
            case '+':password[a]+=num;break;
            case '-':password[a]-=num;break;
            case '*':password[a]*=num;break;
            case '/':password[a]/=num;break;
            case '=':password[a]=num;break;
        }
    }
    printf("%d %d",password[0],password[1]);
    return 0;
}
int comp(const void *a,const void *b)
{
    int i=0,num_1=0,num_2=0;
    char *s1=(char*)a,*s2=(char*)b;
    for(num_1=0,i=0;s1[i]!=' ';i++) num_1=num_1*10+s1[i]-'0';
    for(num_2=0,i=0;s2[i]!=' ';i++) num_2=num_2*10+s2[i]-'0';
    return num_1-num_2;
}
```

G Lake Counting

难度	考点
3	搜索

题目分析

本题可以用深度优先搜索算法（DFS）解决。从任意的 `W` 开始，不停地把邻接的部分用 `.` 代替。1次DFS后与初始的这个 `W` 连接的所有 `W` 就都被替换成了 `.` ，因此直到图中不再存在 `W` 为止，总共进行的DFS的次数就是答案了。

示例代码

```
#include<stdio.h>
#define MAX_N 105
int N,M;
char field[MAX_N][MAX_N+1];//园子
char s[10];//用来处理输入

//现在位置(x,y)
void dfs(int x, int y){
    int dx,dy,nx,ny;
    //将现在所在位置替换为.
```

```
    field[x][y]='.';

    //循环遍历移动的8个方向
    for(dx=-1;dx<=1;dx++){
        for(dy=-1;dy<=1;dy++){
            nx=x+dx,ny=y+dy;
            //判断(nx,ny)是不是在园子内，以及是否有积水
            if(0<=nx&&nx<N&&0<=ny&&ny<M&&field[nx][ny]=='W') dfs(nx,ny);
        }
    }
    return;
}

void solve(){
    int res=0,i,j;
    for(i=0;i<N;i++){
        for(j=0;j<M;j++){
            if(field[i][j]=='W'){
            //从有W的地方开始dfs
            dfs(i,j);
            res++;
            }
        }
    }
    printf("%d",res);
}
int main(){
    int i,j;
    scanf("%d%d",&N,&M);
    gets(s);
    for(i=0;i<N;i++){
        for(j=0;j<M;j++){
            scanf("%c",&field[i][j]);
        }
        gets(s);
    }
    solve();
    return 0;
}
```

H该背单词了

难度	考点
2	qsort，字符指针

题目分析

题意就是对一系列字符串按照指定的法则排序。排序规则为单词长度作为第一关键字升序，单词长度相等的情况下以字典序作为第二关键字升序，根据排序规则写好`qsort`的`cmp`函数即可。这里采用对字符指针排序的方法，将指向原始的每个单词的字符指针单独存成数组，对这个指针数组采用`qsort`排序即可，要注意的就是`cmp`函数的两个`const void*`参数是指向这个数组元素的指针类型，也就是指向`char*`的指针，而并不直接是指向单词首字母(`char`)的指针（直接对字符串数组/字符二维数组用`qsort`也可以）

（非常简单，如果熟练使用`qsort`则相当于签到题难度）

示例代码

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

char words[10005][105];
char *wordptr[10005];
int cnt = 0;

int cmp(const void *a, const void *b) {
    char *pa = *(char**)a, *pb = *(char**)b;
    int la = strlen(pa), lb = strlen(pb);
    if (la != lb) return la < lb ? -1 : 1;
    else return strcmp(pa, pb);
}

int main()
{
    int i = 0;
    while (scanf("%s", words[cnt]) != EOF) {
        wordptr[cnt] = words[cnt];
        cnt++;
    }
    qsort(wordptr, cnt, sizeof(char*), cmp);
    for (i = 0; i < cnt; i++)
        puts(wordptr[i]);

    return 0;
}
```

I 摸摸摸

难度	考点
3	数组

题目分析

两数组两指针往后跑，跑到（m+n）/2即可，另需判断奇偶。

示例代码

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

double findMedianSortedArrays(int* nums1, int nums1Size, int* nums2, int
nums2Size){
    int cernterSite = 0;
    int i = 0;
    int j = 0;
    int k = 0;
    double preValue = 0;
    double curValue = 0;

    cernterSite = (nums1Size + nums2Size)/2;

    for(k = 0; k <= cernterSite; k++){
        if(i < nums1Size && j < nums2Size){
            if(*(nums1 + i) > *(nums2 + j)){
                preValue = curValue;
                curValue = *(nums2 + j);
                j++;
                continue;
            }else{
                preValue = curValue;
                curValue = *(nums1 + i);
                i++;
                continue;
            }
        }

        if(j < nums2Size){
            preValue = curValue;
            curValue = *(nums2 + j);
            j++;
            continue;
        }

        if(i < nums1Size){
            preValue = curValue;
```

```
            curValue = *(nums1 + i);
            i++;
            continue;
        }
    }

    if((nums1Size + nums2Size)%2){
        return curValue;
    }else{
        return (preValue+curValue)/2;
    }
}

int a[100000],b[100000],m,n,i,j;
int main(){
    while (~scanf("%d %d",&m,&n)){
        for (i=0;i<m;i++){
            scanf("%d",&a[i]);
        }
        for (i=0;i<n;i++){
            scanf("%d",&b[i]);
        }
        double p=findMedianSortedArrays(a,m,b,n);
        printf("%.1lf\n",p);
    }
    return 0;
}
```

J Hilbert Curve

难度	考点
4	递归

题目分析

不难发现将 H_n 分成四个区域后，一个格子的编号等于从起点到它所在区域之前经过的格子数，加上从它所在区域的起点到该格子经过的格子数之和。前者因为会完整地经过一个或多个区域，因此格子数是 $2^{n-1} \times 2^{n-1}$ 的倍数；后者的计算相当于在 H_{n-1} 里求某个格子的编号，只要将坐标从 H_n 转换到 H_{n-1} 上后递归计算即可。

示例程序

```
#include <stdio.h>

// 返回 k 阶希尔伯特曲线中，位于 (x, y) 处点的编号
int Idx(int x, int y, int k) {
```

```
    if (x == 1 && y == 1) return 1;
    int p = 1 << (k - 1);
    if (x <= p && y <= p)                // 左上角
        return Idx(y, x, k - 1);
    else if (x > p && y <= p)            // 左下角
        return 1 * p * p + Idx(x - p, y, k - 1);
    else if (x > p && y > p)             // 右下角
        return 2 * p * p + Idx(x - p, y - p, k - 1);
    else                                 // 右上角
        return 3 * p * p + Idx(2 * p - y + 1, p - x + 1, k - 1);
}

int main() {
    int q, n, siz;
    int i, j;

    scanf("%d", &q);
    while (q--) {
        scanf("%d", &n);
        siz = 1 << n;
        for (i = 1; i <= siz; i++)
            for (j = 1; j <= siz; j++)
                printf("%d%c", Idx(i, j, n), " \n"[j == siz]);
    }

    return 0;
}
```

K 谁是19王 Round 2

难度	考点
3	简单博弈

题目分析

显然，如果最后面对的数字是1，则没有办法进行操作。易知，奇数的因子一定是奇数，而两个奇数相减得到的一定是偶数。故如果游戏开始时 n 为偶数，YangHui只需要一直选1，使秋月面临 n 为奇数的情况，到最后一定能胜利；反之，若n为奇数，YangHui无论选择什么，n 都会变成偶数，与前文同理，最后秋月必胜。
注：不知道大家看懂HINT了没，这个题如果你只输出`YangHui`或`Suzumiya`都是能得到分数的，有些题目是会给大家一些骗分的机会的，大家可以感受一下，有时这也不失为一种考试技巧。

参考代码

```c
#include<stdio.h>
int main()
{
    int q, n;
    scanf("%d", &q);
    while(q--)
    {
        scanf("%d", &n);
        printf("%s", n & 1 ? "Suzumiya\n" : "YangHui\n");
    }
    return 0;
}
```

LYourSQL

难度	考点
6	结构、联合、排序

题目解析

只是一道使用结构体排序的题目。主要难点可能在于数据存储以及`compare`函数的设计，另外不同数据之间如何进行比较也值得注意。具体做法请仔细阅读示例代码。

示例代码

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define ROWNUM 1007
#define COLNUM 107
#define LEN 107
#define INT 0
#define REAL 1
#define VARCHAR 2
#define DATE 3
int type[COLNUM], sp[COLNUM][2], row, col, n;
char title[COLNUM][LEN];
typedef struct table
{
    union attr
    {
        char varchar[LEN];
        double real;
        int date;
        int intNum;
```

```
    } A;
    char rawData[LEN];
} T;
T t[ROWNUM][COLNUM];

void readTable()
{
    int i, j, y, m, d;
    char typeString[LEN];
    scanf("%d %d", &row, &col);
    for (i = 0; i < col; i++)
        scanf("%s", title[i]);
    for (i = 0; i < col; i++)
    {
        scanf("%s", typeString);
        if (strcmp(typeString, "INT") == 0)
            type[i] = INT;
        else if (strcmp(typeString, "REAL") == 0)
            type[i] = REAL;
        else if (strcmp(typeString, "VARCHAR") == 0)
            type[i] = VARCHAR;
        else
            type[i] = DATE;
    }
    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
        {
            scanf("%s", t[i][j].rawData);
            switch (type[j])
            {
                case INT:
                    sscanf(t[i][j].rawData, "%d", &t[i][j].A.intNum);
                    break;
                case REAL:
                    sscanf(t[i][j].rawData, "%lf", &t[i][j].A.real);
                    break;
                case VARCHAR:
                    sscanf(t[i][j].rawData, "%s", t[i][j].A.varchar);
                    break;
                case DATE:
                    sscanf(t[i][j].rawData, "%d-%d-%d", &y, &m, &d);
                    t[i][j].A.date = d + m * 100 + y * 10000;
                default:
                    break;
            }
        }
}

void readSP()
```

```
{
    int i;
    char colName[LEN];
    while (~scanf("%s %d", colName, &sp[n][1]))
    {
        for (i = 0; i < col; i++)
            if (strcmp(colName, title[i]) == 0)
            {
                sp[n][0] = i;
                break;
            }
        n++;
    }
}

int cmp(const void *a, const void *b)
{
    T *t1 = (T *)a, *t2 = (T *)b;
    int i, c, sgn, ret = 0;
    for (i = 0; i < n; i++)
    {
        c = sp[i][0];
        sgn = sp[i][1];
        switch (type[c])
        {
            case VARCHAR:
                ret = sgn * strcmp(t1[c].A.varchar, t2[c].A.varchar);
                break;
            case INT:
                ret = sgn * (t1[c].A.intNum - t2[c].A.intNum);
                break;
            case REAL:
                if (t1[c].A.real - t2[c].A.real > 0)
                    ret = sgn * 1;
                else if (t1[c].A.real - t2[c].A.real < 0)
                    ret = sgn * -1;
                else
                    ret = 0;
                break;
            case DATE:
                ret = sgn * (t1[c].A.date - t2[c].A.date);
            default:
                break;
        }
        if (ret != 0)
            return ret;
    }
    return 0;
}
```

```
void printTable()
{
    int i, j;
    for (j = 0; j < col; j++)
        printf("%s ", title[j]);
    putchar('\n');
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
            printf("%s ", t[i][j].rawData);
        putchar('\n');
    }
}

int main(int argc, const char * argv[])
{
    readTable();
    readSP();
    qsort(t, row, sizeof(*t), cmp);
    printTable();
    return 0;
}
```