C4 - Solution

A - 成绩分段打印

难度	考点
1	if-else语句

题目分析

直接按照题目中的要求分类讨论即可。

参考代码

```c
#include<stdio.h>
int main(){
    int score;
    scanf("%d",&score);
    if(90<=score&&score<=100)
        printf("A");
    else if(80<=score&&score<90)
        printf("B");
    else if(70<=score&&score<80)
        printf("C");
    else if(60<=score&&score<70)
        printf("D");
    else
        printf("F");
    return 0;
}
```

B - 天数

难度	考点
1	判断

题目分析

根据年份判断闰年，闰年二月为29天。

注意多组数据。

示例代码

```c
#include <stdio.h>
int main()
{
    int T, y, m;
    int a[13] = {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int b[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d%d", &y, &m);
        if ((y % 4 == 0) && (!(y % 100 == 0)) || (y % 400 == 0))
            printf("%d\n", a[m]);
        else
            printf("%d\n", b[m]);
    }
    return 0;
}
```

C - 一天不用switch我浑身难受

难度	考点
1	switch语句

题目分析

依照题意，将给定代码的 `if-else` 格式换成 `switch` 格式即可。

其他的都不用管！

别忘了break！

示例代码

```
#include<stdio.h>
unsigned seed;
long long a, b;
long long ans = 0;
int n, i;
int op;
int scheme[35];
int main() {
    scanf("%d%u", &n, &seed);
    for (i = 0; i <= 31; ++i) scanf("%d", &scheme[i]);
    while(n--) {
                seed ^= seed << 13;
                seed ^= seed >> 17;
                seed ^= seed << 5;
                op = scheme[seed & 31] & 31;
                a = n ^ 114514191981011;
                b = n ^ 191981011451411;
                switch(op){
                        case 0 : ans += a + b; break;
                        case 1: ans += a - b; break;
                        case 2: ans += a * b; break;
                        case 3: ans += a / b; break;
                        case 4: ans += a % b; break;
                        case 5: ans += a & b; break;
                        case 6: ans += a | b; break;
                        case 7: ans += a ^ b; break;
                        case 8: ans -= a ^ b; break;
                        case 9: ans -= a | b; break;
                        case 10: ans -= a & b; break;
                        case 11: ans -= a % b; break;
                        case 12: ans -= a / b; break;
                        case 13: ans -= a * b; break;
                        case 14: ans -= a - b; break;
                        case 15: ans -= a + b; break;
                        case 16: ans &= a * b; break;
                        case 17: ans &= a / b; break;
                        case 18: ans &= a + b; break;
                        case 19: ans &= a - b; break;
                        case 20: ans &= a | b; break;
                        case 21: ans &= a ^ b; break;
                        case 22: ans &= a % b; break;
                        case 23: ans &= a & b; break;
                        case 24: ans ^= a / b; break;
                        case 25: ans ^= a * b; break;
                        case 26: ans ^= a - b; break;
                        case 27: ans ^= a + b; break;
                        case 28: ans ^= a ^ b; break;
                        case 29: ans ^= a | b; break;
                        case 30: ans ^= a & b; break;
                        case 31: ans ^= a % b; break;
                }
    }
    printf("%lld", ans);
    return 0;
}
```

D - 成绩分析

难度	考点
2	判断

题目分析

需要统计三个班的最高分，最低分和平均分。

分别用数组存下最高分（Max）、最低分（Min）、班级分数总和（sum）、班级人数（Num）。

对于N名同学，每输入一个人的信息，就更新统计信息；

最后统计平均分时，由于数据范围小，相乘不会溢出，可以转为double类型来计算平均分，也可以将$\frac{sum_A}{num_A} < \frac{sum_B}{num_B}$转为$sum_A \times num_B < sum_B \times num_A$来做。

示例代码

```c
#include<stdio.h>
int main()
{
	int N,Max[3],Min[3],sum[3],num[3];
	int clas,scor;
	for(int i=0;i<=2;i++){//初始化
		Max[i]=-1;
		Min[i]=101;
		sum[i]=0;
		num[i]=0;
	}
	scanf("%d",&N);
	for(int i=0;i<N;i++){
		scanf(" %c%d",&clas,&scor);//注意%c前的空格，用于吃掉空白符（换行）
		clas-='A';	//A->0	B->1	C->2
		Max[clas]=scor>Max[clas]?scor:Max[clas];//统计最大最小的分数
		Min[clas]=scor<Min[clas]?scor:Min[clas];
		sum[clas]+=scor;//分数求和
		num[clas]++;//统计人数
	}

	int max_ban=0;//计算哪个班的平均分最大，考虑到不会溢出，不用double计算了
	if(sum[max_ban]*num[1]<sum[1]*num[max_ban]){
		max_ban=1;
	}
	if(sum[max_ban]*num[2]<sum[2]*num[max_ban]){
		max_ban=2;
	}
	printf("%c\n",'A'+max_ban);
	printf("%d %d\n",Max[0],Min[0]);
	printf("%d %d\n",Max[1],Min[1]);
	printf("%d %d",Max[2],Min[2]);
	return 0;
}
```

E - Black and White

难度	考点
2	循环、判断

题目分析

可以设一个变量cnt来表示当前遇到的字母，0表示w，1表示b。

如果当前遇到的字母是w并且cut为1，那么说明在b的后面还有额外的字母w，是不可爱的，直接跳出循环并输出`No`。

如果扫完了整个字符串之后没有发现上述情况，那么输出`Yes`即可。

示例代码

```c
#include <stdio.h>
int main()
{
    char c;
    int cnt = 0, flag = 1;
    while ((c = getchar()) != EOF)
    {
        if (c == 'w')
        {
            if (cnt == 1)
            {
                flag = 0;
                break;
            }
        }
        if (c == 'b')
            cnt = 1;
    }
    puts(flag == 0 ? "No" : "Yes");
    return 0;
}
```

F - 愿此行，终抵群星

难度	考点
2	循环、数学

题目分析

如果一个点集能用关于x的不超过$n-1$次的多项式表示，那么点集中一定没有两个点的横坐标相同。

下面是简要地证明：

将n个点的坐标带入到函数$f(x)=\sum_{i=0}^{n-1}a_ix^i$中，可以得到$n$个方程。

由于我们现在要求的是系数a_i，所以将所有x^i提取出来组成行列式，就得到了范德蒙行列式，即：

形如：

$$|D_n|_{(n\geq 2)}=\begin{vmatrix}1&1&1&\cdots&1\\x_1&x_2&x_3&\cdots&x_n\\x_1^2&x_2^2&x_3^2&\cdots&x_n^2\\\cdots&\cdots&\cdots&\cdots&\cdots\\x_1^{n-2}&x_2^{n-2}&x_3^{n-2}&\cdots&x_n^{n-2}\\x_1^{n-1}&x_2^{n-1}&x_3^{n-1}&\cdots&x_n^{n-1}\end{vmatrix}\text{或}\begin{vmatrix}1&x_1&x_1^2&\cdots&x_1^{n-2}&x_1^{n-1}\\1&x_2&x_2^2&\cdots&x_2^{n-2}&x_2^{n-1}\\1&x_3&x_3^2&\cdots&x_3^{n-2}&x_3^{n-1}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\1&x_{n-1}&x_{n-1}^2&\cdots&x_{n-1}^{n-2}&x_{n-1}^{n-1}\\1&x_n&x_n^2&\cdots&x_n^{n-2}&x_n^{n-1}\end{vmatrix}$$

称为范德蒙行列式。（线性代数后面会学到这个重要的行列式）

而范德蒙行列式有个非常好的性质：

$$|D_n|=\prod_{1\leq j<i\leq n}(x_i-x_j)$$

我们知道如果行列式的值不为0，那么原方程组有解。所以由范德蒙行列式的这个性质我们可以推出：只要横坐标没有相同的，原方程一定有解。

示例代码

```c
#include<stdio.h>
#define MAXN 1010
int n,flag,x[MAXN],y[MAXN];
int main(){
    int t,i,j,cases;
    scanf("%d",&t);
    for(cases=1;cases<=t;cases++){
        flag=1;
        scanf("%d",&n);
        for(i=1;i<=n;i++)
            scanf("%d%d",&x[i],&y[i]);
```

```
        for(i=2;i<=n&&flag;i++)
            if(x[i]==x[i-1])
                flag=0;//用flag表示是否有重复的横坐标，有重复则为0，否则为1
        printf("Case #%d: ",cases);
        printf(flag?"Through the star sea.\n":"Stop somewhere.\n");//使用三目运算符来简化输出
    }
    return 0;
}
```

G - 炸弹人

难度	考点
4	循环控制

题目分析

根据提示所述，用外层循环枚举敌人，用内层循环枚举炸弹，统计该敌人是否被轰炸。具体来说，如果敌人i与炸弹j的横坐标或者纵坐标相同，则最终得分sum加上s_i,并且退出内层循环(`break`语句)。（因为这个敌人已经确定被轰炸，如果不立刻退出，可能造成重复计数）。

记得sum开`long long`！！！

示例代码

```
#include<stdio.h>
#define N 1005
int n,m;
int px[N],py[N],s[N],qx[N],qy[N];
int main()
{
	scanf("%d%d",&n,&m);
	for(int i=1;i<=n;i++)
		scanf("%d%d%d",&px[i],&py[i],&s[i]);
	for(int i=1;i<=m;i++)
		scanf("%d%d",&qx[i],&qy[i]);

	long long sum = 0;
	for(int i=1;i<=n;i++)
	{
		for(int j=1;j<=m;j++)
		{
			if(px[i] == qx[j] || py[i] == qy[j])//横坐标相等，或纵坐标相等
			{
				sum += s[i];
				break;//break语句退出最里面的一层循环
			}
		}
	}
	printf("%lld",sum);
	return 0;
}
```

H - 有效字符

难度	考点
2	字符串

题目分析

题目要求找到有效字符数**最多**的语句，因此需要用一些变量来存储答案：sen记录语句内容，max记录该语句的有效字符数，ans记录是第几条语句。

对于有效字符的判断，可以使用 $ctype.h$ 中的函数 $isalnum$ ，该函数可判断字符是否是数字或字母。

因为待处理的字符串只有一行，所以可以先整行读取。在遍历的同时用变量 tmp 、num 、len 、cnt 分别记录当前语句的内容、有效字符数、长度和是第几条语句。当遇到语句的末尾符 (.，!，?) 时，将当前语句的有效字符数和已经记录的最大值进行比较，如果大于（题目中说两语句有效字符数相同时取前面的语句，因此不能用大于等于）则进行相关值的替换。

需要注意的一点是，根据题目中对于语句的定义，只有当出现有效字符时才能算是一条语句的开始，因此先得跳过所有的无效字符。并且在记录语句个数的时候，不能只以是否出现末尾符为判断条件，而要考虑是否有完整的格式（以有效字符开头，以末尾符结尾）。

示例代码

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

char s[1000005], sen[1000005], tmp[1000005];

int main()
{
	int flag = 0, cnt = 1, num = 0, max = 0, len = 0, ans; //各变量的含义在题目分析中
	gets(s);
	for (int i = 0; s[i]; i ++)   //在条件判断时s[i]相当于s[i] != 0，因此在遇到字符串结尾符'\0'时会退出循环
	{                                                                    //flag表示是否在一条语句当中
		if (!flag && !isalnum(s[i])) continue;       //若flag为0并且还未遇到有效字符，则跳过当前字符
		if (!flag) flag = 1, len = 0, num = 0;       //若flag为0并且遇到了有效字符，则表示进入了一条新语句，将flag改为1，并
		tmp[len ++] = s[i];
		if (isalnum(s[i])) num ++;
		if (s[i] == '.' || s[i] == '!' || s[i] == '?')
		{
			if (num > max)
			{
				max = num;
				ans = cnt;
				tmp[len] = '\0';       //要将字符串结尾用'\0'封上，否则无法进行拷贝
				strcpy(sen, tmp);      //本题实际上不需要进行字符串的拷贝，大家可以想一下要怎么做
			}
			flag = 0, cnt ++;              //一条语句结束时要把flag重新改为0，并且将语句个数加1
		}
	}
	printf("%d\n%d\n%s", ans, max, sen);
	return 0;
}
```

I - Monica的羊

难度	考点
5	模拟，标记数组

题目分析

阅读题面，模拟这个操作。

使用一个标记数组表示每一只羊是否出圈，具体的$vis[i]=1$表示编号为i的羊已经出圈，$vis[i]=0$表示还未出圈。

由于会出圈$n-1$次，故循环$n-1$次，每次循环查找哪些羊未出圈，同时计数，数量达到k时$break$跳出循环，标记出圈即可。

示例代码

```c
#include<stdio.h>
#define MN (10000+5)
int n,k;
int vis[MN];
int main(){
	scanf("%d%d",&n,&k);
	int p=n-1;
	for(int i=1;i<n;++i){
		int cnt=0;
		while(1){
			if(!vis[p])cnt++;
			if(cnt==k)break;
			p=p%n+1;
		}
		vis[p]++;
	}
	for(int i=1;i<=n;++i)
```

```
                if(!vis[i])
                        printf("%d\n",i);
        return 0;
}
```

J - HDT

难度	考点
6	模拟

题目分析

按照题目要求模拟即可。

一些实现细节见代码。

示例代码

```
#include <stdio.h>
unsigned long Next = 1;
int hp[2][8], atk[2][8], num[2], p[2], death[2];
// 0代表玩家A，1代表玩家B
// num[0]表示玩家A初始时拥有的随从数量，num[1]表示表示玩家B初始时拥有的随从数量
// p[0]表示当前玩家A应该发起攻击的随从，p[1]表示当前玩家B应该发起攻击的随从
// death[0]表示玩家A死亡的随从数量，death[1]表示玩家B死亡的随从数量
int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        //读入部分（步骤1）
        death[0] = death[1] = 0;
        p[0] = p[1] = 1;
        scanf("%d", &num[0]);
        for (int i = 1; i <= num[0]; i++)
            scanf("%d %d", &atk[0][i], &hp[0][i]);
        scanf("%d", &num[1]);
        for (int i = 1; i <= num[1]; i++)
            scanf("%d %d", &atk[1][i], &hp[1][i]);

        //战斗部分（步骤2～5）
        for (int now = 0;; now ^= 1) // 这里用now表示攻击方（A/B），使用异或运算可以方便地更换攻击方，以及表示被攻击方（即now^1）(这里定义0表示A，1表示B)
        {
            // 判定游戏是否结束（步骤2）
            int f = ((num[0] == death[0]) << 1) | (num[1] == death[1]); // 这里使用f来标志当前随从存活状态，其中二进制下00表示AB均有随从存活，01表示B无随从存活，10表示A无随从存活，11表示AB均无随从
            if (f)
            {
                puts(f == 3 ? "Draw!" : (f == 2 ? "B Win!" : "A Win!"));// 使用三目运算符判断要输出的字符串
                break;
            }
            // 判定发起攻击的随从（步骤3）
            while (hp[now][p[now]] == 0)
                p[now] = (p[now] == num[now]) ? 1 : p[now] + 1; //使用三目运算符判断p[now]是否到达最右边的随从
            // 判定被攻击的随从（步骤3）
            Next = Next * 810975 + 922768;
            int cnt = ((unsigned)(Next / 65536) % (num[now ^ 1] - death[now ^ 1]) + 1), goal = 0;
            while (cnt)
            {
                if (hp[now ^ 1][++goal])
                    cnt--;
            }
            // 战斗阶段（步骤4）
            hp[now][p[now]] -= atk[now ^ 1][goal], hp[now ^ 1][goal] -= atk[now][p[now]];
            if (hp[now ^ 1][goal] <= 0)
                hp[now ^ 1][goal] = 0, death[now ^ 1]++;
            if (hp[now][p[now]] <= 0)
                hp[now][p[now]] = 0, death[now]++;
            // 进入下一轮（步骤5）
            p[now] = (p[now] == num[now]) ? 1 : p[now] + 1;
        }
    }
```

```
    return 0;
}
```