继续注意点和试题选讲

—— 郑任毅

重载小于号和 sort 续讲

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return a < x.a;
    }
};
```

重载小于号和 sort 续讲

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return a < x.a;
    }
};
```

还是这个问题，重载的函数体里的语句，究竟是什么意思?

我们认为重载函数的函数体里的 a 、 b 这样的基本变量属于 node

那么这个函数可以这样理解：这个 node 里的变量（ a 、 b ）和另一个 node x 里的（ x.a 、 x.b ）相比，满足什么样的条件时，这个 node 应该排在前面，或者说这个 node 算是小的

那么实际上也就是满足我们 return 的条件时，这个 node 排在前面

重载小于号和 sort 续讲

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return a < x.a;
    }
};
```

也就是说：

if(this.a < x.a) 那么编译器认为 this < x，在从小到大的排序中， this 排在 x 前面

所以在这种重载之下，谁的 a 小，谁就小，因此实现了按照关键字 a 从小到大排序。

重载小于号和 sort 续讲

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return a > x.a;
    }
};
```

反之，如上重载：

if(this.a > x.a) 那么编译器认为 this < x，在从小到大的排序中， this 排在 x 前面

所以在这种重载之下，谁的 a 大，谁就小，因此实现了按照关键字 a 从大到小排序。

重载小于号和 sort 续讲

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return ???
    }
}
```

- 最后重申，在这种重载小于号的方法中
  - 按 a 从小到大排： return a < x.a;
  - 按 a 从大到小排： return a > x.a;
  - 按 b 从小到大排： return b < x.b;
  - 按 a 和 b 的比值从小到大排： return a/b < x.a/x.b;
    - 可按需要换成交叉相乘

例题 B1015 德才论

- 若干废话……现给出一批考生的德才分数，请根据司马光的理论给出录取排名。
- 输入格式：
- 输入第 1 行给出 3 个正整数，分别为： N （<=105），即考生总数；L （>=60），为录取最低分数线，即德分和才分均不低于 L 的考生才有资格被考虑录取； H （<100），为优先录取线——德分和才分均不低于此线的被定义为“才德全尽”，此类考生按德才总分从高到低排序；才分不到但德分到线的一类考生属于“德胜才”，也按总分排序，但排在第一类考生之后；德才分均低于 H，但是德分不低于才分的考生属于“才德兼亡”但尚有“德胜才”者，按总分排序，但排在第二类考生之后；其他达到最低线 L 的考生也按总分排序，但排在第三类考生之后。
- 随后 N 行，每行给出一位考生的信息，包括：准考证号、德分、才分，其中准考证号为 8 位整数，德才分为区间 [0, 100] 内的整数。数字间以空格分隔。

例题 B1015 德才论

- 输出格式：
- 输出第 1 行首先给出达到最低分数线的考生人数 M，随后 M 行，每行按照输入格式输出一位考生的信息，考生按输入中说明的规则从高到低排序。当某类考生中有多人总分相同时，按其德分降序排列；若德分也并列，则按准考证号的升序输出。

例题 B1015 德才论

- 思路：又是熟悉的第五题，明显又是花式排序

例题 B1015 德才论

- 思路：又是熟悉的第五题，明显又是花式排序
- 首先需要输出的实际上分四类：
  - 德 >= H ，才 >= H
  - 德 >= H ， L <= 才 < H
  - L <= 德 < H， L <= 才 < H，德 >= 才
  - L <= 德 < H， L <= 才 < H，德 < 才

例题 B1015 德才论

- 思路：又是熟悉的第五题，明显又是花式排序
- 首先需要输出的实际上分四类：
  - 德 >= H ，才 >= H
  - 德 >= H ， L <= 才 < H
  - L <= 德 < H， L <= 才 < H，德 >= 才
  - L <= 德 < H， L <= 才 < H，德 < 才
- 我们可以发现，其实在四类考生的递进中，条件可以简化

例题 B1015 德才论

- 我们可以发现，其实在四类考生的递进中，条件可以简化：
  - 德 >= H ，才 >= H
  - 德 >= H ，才 >= L
  - 德 >= L，才 >= L，德 >= 才
  - 德 >= L，才 >= L

例题 B1015 德才论

- 我们可以发现，其实在四类考生的递进中，条件可以简化：
  - 德 >= H ，才 >= H
  - 德 >= H ，才 >= L
  - 德 >= L，才 >= L，德 >= 才
  - 德 >= L，才 >= L
- 根据以上条件，我们可以将每个考生定义成一个结构体，然后将符合上述条件的考生分放进四个结构体数组中

例题 B1015 德才论

- 根据以上条件，我们可以将每个考生定义成一个结构体，然后将符合上述条件的考生分放进四个结构体数组中

```
struct people {
  int id, a, b;      // 考生证号、德分、才分
}
lv1[maxn],lv2[maxn],lv3[maxn],lv4[maxn];
```

例题 B1015 德才论

- 德 >= H ，才 >= H
- 德 >= H ，才 >= L
- 德 >= L，才 >= L，德 >= 才
- 德 >= L，才 >= L

```
for (int i = 1; i <= n; ++i) {
    people tmp;
    scanf("%d%d%d", &tmp.id, &tmp.a, &tmp.b);
    if (tmp.a >= h && tmp.b >= h )lv1[++cnt1] = tmp;
    else if (tmp.a >= h && tmp.b >= l )lv2[++cnt2] = tmp;
    else if (tmp.a >= l && tmp.b >= l && tmp.a >=
tmp.b )lv3[++cnt3] = tmp;
    else if (tmp.a >= l && tmp.b >= l )lv4[++cnt4] = tmp;
  }
```

例题 B1015 德才论

- 分发完考生之后，其实就是对于每一个档次的考生进行排序了，那么我们重新研究一下排序要求
- 考生按输入中说明的规则（德才总分）从高到低排序。当某类考生中有多人总分相同时，按其德分降序排列；若德分也并列，则按准考证号的升序输出。

例题 B1015 德才论

- 分发完考生之后，其实就是对于每一个档次的考生进行排序了，那么我们重新研究一下排序要求
- 考生按输入中说明的规则（德才总分）从高到低排序。当某类考生中有多人总分相同时，按其德分降序排列；若德分也并列，则按准考证号的升序输出。
  - 按总分从高到低排序
  - 总分相同，按德分从高到低排序
  - 德分也相同，按准考证号从低到高排序

sort 的自由定制：多关键字排序

- 当我们对一个结构体排序时，常常会遇到，需要对多个关键字排序，这些关键字分主次
- 优先按第一关键字如何排序
- 在第一关键字相等时再按第二关键字排序
- 在第二关键字相等时再按第三关键字排序
- ……
- 以此类推

sort 的自由定制：多关键字排序

- 多关键字排序同样可以通过重载小于号来实现！

sort 的自由定制：多关键字排序

- 多关键字排序同样可以通过重载小于号来实现！

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return ???
    }
};
```

- 仍然以此为例，假设我们要先按 a 升序（从小到大），再按 b 降序（从大到小）

sort 的自由定制：多关键字排序

- 仍然以此为例，假设我们要先按 a 升序（从小到大），再按 b 降序（从大到小）

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return ???
    }
};
```

- 考虑我们之前说的， this 的属性和 x 的属性满足什么关系的时候， this 排名靠前?

sort 的自由定制：多关键字排序

- 仍然以此为例，假设我们要先按 a 升序（从小到大），再按 b 降序（从大到小）

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        return ???
    }
};
```

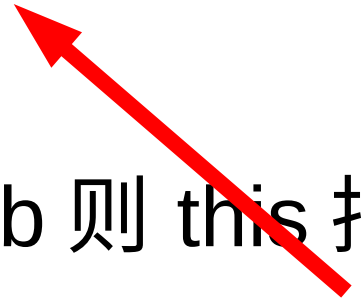
- 当 a == x.a 时， b > x.b 则 this 排名靠前
- 否则 a < x.a 时， this 排名靠前

sort 的自由定制：多关键字排序

- 仍然以此为例，假设我们要先按 a 升序（从小到大），再按 b 降序（从大到小）

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        if(a == x.a) return b > x.b;
        return a < x.a;
    }
};
```

- 当 a == x.a 时， b > x.b 则 this 排名靠前
- 否则 a < x.a 时， this 排名靠前



sort 的自由定制：多关键字排序

```
struct node{
    int a,b;
    bool operator < (const node x)const{
        if(a == x.a) return b > x.b;
        return a < x.a;
    }
};
```

- 那么仍然给出结论：
  - 越高优先级的关键字，其比较的 return 语句越靠后
  - 低优先级的关键字，其进行比较的条件是所有较高优先级的关键字都无法比较出结果（都相等）
  - return 的内容与普通的排序重载一样（按谁比谁、从小到大 < 、从大到小 > ）

例题 B1015 德才论

- 按总分从高到低排序
- 总分相同，按德分从高到低排序
- 德分也相同，按准考证号从低到高排序

```
struct people {
  int id, a, b;      // 考生证号、德分、才分
};
```

- 按 (a + b) 从高到低排序
- (a + b) 相同，按 a 从高到低排序
- a 也相同，按 id 从低到高排序

例题 B1015 德才论

- 按 (a + b) 从高到低排序
- (a + b) 相同，按 a 从高到低排序
- a 也相同，按 id 从低到高排序

- 越高优先级的关键字，其比较的 return 语句越靠后
- 低优先级的关键字，其进行比较的条件是所有较高优先级的关键字都无法比较出结果（都相等）
- return 的内容与普通的排序重载一样（按谁比谁、从小到大 < 、从大到小 > ）

```
bool operator < (const node x)const{
    if (a + b == x.a + x.b&&a == x.a)return id < x.id;
    if (a + b == x.a + x.b)return a > x.a;
    return a + b > x.a + x.b;
}
```

例题 B1015 德才论

```
sort(lv1 + 1, lv1 + cnt1 + 1);
sort(lv2 + 1, lv2 + cnt2 + 1);
sort(lv3 + 1, lv3 + cnt3 + 1);
sort(lv4 + 1, lv4 + cnt4 + 1);

printf("%d\n", cnt1 + cnt2 + cnt3 + cnt4);
for (int i = 1; i <= cnt1; ++i) {
    printf("%08d %d %d\n", lv1[i].id, lv1[i].a, lv1[i].b);
}
for (int i = 1; i <= cnt2; ++i) {
    printf("%08d %d %d\n", lv2[i].id, lv2[i].a, lv2[i].b);
}
for (int i = 1; i <= cnt3; ++i) {
    printf("%08d %d %d\n", lv3[i].id, lv3[i].a, lv3[i].b);
}
for (int i = 1; i <= cnt4; ++i) {
    printf("%08d %d %d\n", lv4[i].id, lv4[i].a, lv4[i].b);
}
```

例题 1035 插入与归并

- 根据维基百科的定义：
- 插入排序是迭代算法，逐一获得输入数据，逐步产生有序的输出序列。每步迭代中，算法从输入序列中取出一元素，将之插入有序序列中正确的位置。如此迭代直到全部元素有序。
- 归并排序进行如下迭代操作：首先将原始序列看成 N 个只包含 1 个元素的有序子序列，然后每次迭代归并两个相邻的有序子序列，直到最后只剩下 1 个有序的序列。
- 现给定原始序列和由某排序算法产生的中间序列，请你判断该算法究竟是哪种排序算法？

例题 1035 插入与归并

- 输入格式:
- 输入在第一行给出正整数 N (<=100)；随后一行给出原始序列的 N 个整数；最后一行给出由某排序算法产生的中间序列。这里假设排序的目标序列是升序。数字间以空格分隔。
- 输出格式:
- 首先在第 1 行中输出“ Insertion Sort”表示插入排序、或“ Merge Sort”表示归并排序；然后在第 2 行中输出用该排序算法再迭代一轮的结果序列。题目保证每组测试的结果是唯一的。数字间以空格分隔，且行末不得有多余空格。

例题 1035 插入与归并

- 输入样例 1：
- 10
- 3 1 2 8 7 5 9 4 6 0
- 1 2 3 7 8 5 9 4 6 0
- 输出样例 1：
- Insertion Sort
- 1 2 3 5 7 8 9 4 6 0

- 输入样例 2：
- 10
- 3 1 2 8 7 5 9 4 0 6
- 1 3 2 8 5 7 4 9 0 6
- 输出样例 2：
- Merge Sort
- 1 2 3 8 4 5 7 9 0 6

例题 1035 插入与归并

- 输入样例 1：
- 10
- 3 1 2 8 7 5 9 4 6 0
- 1 2 3 7 8 5 9 4 6 0
- 输出样例 1：
- Insertion Sort
- 1 2 3 5 7 8 9 4 6 0

- 输入样例 2：
- 10
- 3 1 2 8 7 5 9 4 0 6
- 1 3 2 8 5 7 4 9 0 6
- 输出样例 2：
- Merge Sort
- 1 2 3 8 4 5 7 9 0 6

如何判断是什么排序?

例题 1035 插入与归并

- 输入样例 1：
- 10
- 3 1 2 8 7 5 9 4 6 0
- 1 2 3 7 8 5 9 4 6 0
- 输出样例 1：
- Insertion Sort
- 1 2 3 5 7 8 9 4 6 0

- 输入样例 2：
- 10
- 3 1 2 8 7 5 9 4 0 6
- 1 3 2 8 5 7 4 9 0 6
- 输出样例 2：
- Merge Sort
- 1 2 3 8 4 5 7 9 0 6

如何判断是什么排序?

例题 1035 插入与归并

- 输入样例 1：
- 10
- 3 1 2 8 7 5 9 4 6 0
- 1 2 3 7 8 5 9 4 6 0
- 输出样例 1：
- Insertion Sort
- 1 2 3 5 7 8 9 4 6 0

- 输入样例 2：
- 10
- 3 1 2 8 7 5 9 4 0 6
- 1 3 2 8 5 7 4 9 0 6
- 输出样例 2：
- Merge Sort
- 1 2 3 8 4 5 7 9 0 6

插入排序前一半有序，后一半与原序列相同

归并排序每 2^k 为一组，组内有序

例题 1035 插入与归并

- 我的做法是，对 a 、 b 数列比较后缀，找到从后往前第一个不一样的地方，再向前判断是否前面的部分有序
  - 若有公共后缀，且前面的部分有序，则是插排
  - 否则是归并排序

例题 1035 插入与归并

```
bool flag1 = 0,flag2 = 0;
int pos = 0;
for(int i = n ; i >= 1 ; --i){
    if(flag1 == 0){
        if(a[i] != b[i])flag1 = 1, pos = i + 1;
    }
    else{
        if(b[i] > b[i+1])flag2 = 1;
    }
}
```

例题 1035 插入与归并

- 分类完之后，各自怎么进行下一轮迭代?

例题 1035 插入与归并

- 分类完之后，各自怎么进行下一轮迭代?
- 3 1 2 8 7 5 9 4 6 0
- 1 2 3 7 8 5 9 4 6 0
- 1 2 3 5 7 8 9 4 6 0
- 插入排序：将后面与原串相同的部分中，第一个数插进前面有序的部分中。
- 即将前 pos 个数进行了排序!
- sort(a + 1, a + 1 + pos); //sort(a, a + pos);

例题 1035 插入与归并

- 分类完之后，各自怎么进行下一轮迭代?
- 3 1 2 8 7 5 9 4 0 6
- 1 3 2 8 5 7 4 9 0 6
- 1 2 3 8 4 5 7 9 0 6
- 归并排序：得到这一轮的每组大小是 2^k (即判断出，每组大小 2^(k+1) 时不满足每组组内有序)
- 对原序列的每个 2^(k+1) 大小的组分别进行排序
- 对每个开始位置分别调用 sort

例题 1035 插入与归并

```
for(int i = 2 ; i <= n ; i <<= 1){
    flag1 = 0;
    for(int j = 1 ; j <= n ; j += i){
        for(int k = j + 1 ; k <= min(j+i-1, n) ; ++k){
            if(b[k] < b[k-1])flag1 = 1;
        }
    }
    if(flag1){
        for(int j = 1 ; j <= n ; j += i){
            int len = min(i, n-j+1);
            sort(a + j, a + j + len);
        }
        break;
    }
}
```

例题 1035 插入与归并

时间	结果	得分	题目
9月09日 00:28	部分正确	24	1035

测试点

测试点	结果	用时(ms)
0	答案正确	4
1	答案正确	4
2	答案错误	6
3	答案正确	2
4	答案正确	3
5	答案正确	3
6	答案正确	4

又来?

为啥?

例题 1035 插入与归并

- 重新考虑插入排序的这样一组样例：
- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0

例题 1035 插入与归并

- 重新考虑插入排序的这样一组样例：
- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 以我的做法，我认为后四个是与原串相同的，而前五个是已经有序的

例题 1035 插入与归并

- 重新考虑插入排序的这样一组样例：
- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 以我的做法，我认为后四个是与原串相同的，而前五个是已经有序的
- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 但是否也可以认为后三个是与原串相同的，前六个是已经有序的呢?

例题 1035 插入与归并

- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 但是否也可以认为后三个是与原串相同的，前六个是已经有序的呢?
- 结果是肯定的，并且我将代码逻辑改为对插入排序的部分寻找一个最长上升前缀，而第一个不满足的作为我下一次迭代需要排序的元素，就能通过

例题 1035 插入与归并

- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 3 1 2 8 7 9 4 6 0
- 1 2 3 7 8 9 4 6 0
- 那么简单的说，这是一道错题，至少数据是有问题的
- 因为题面说保证只有唯一解，并且没有对这样的情况给出具体的处理方法
- 但插入排序在这两轮中序列是相同的，这很普遍

例题 1039 到底买不买

- 小红想买些珠子做一串自己喜欢的珠串。卖珠子的摊主有很多串五颜六色的珠串，但是不肯把任何一串拆散了卖。于是小红要你帮忙判断一下，某串珠子里是否包含了全部自己想要的珠子？如果是，那么告诉她有多少多余的珠子；如果不是，那么告诉她缺了多少珠子。

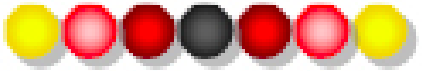
例题 1039 到底买不买

- 为方便起见，我们用 [0-9]、 [a-z]、 [A-Z] 范围内的字符来表示颜色。例如在图 1 中，第 3 串是小红想做的珠串；那么第 1 串可以买，因为包含了全部她想要的珠子，还多了 8 颗不需要的珠子；第 2 串不能买，因为没有黑色珠子，并且少了一颗红色的珠子。

ppRYYGrrYBR2258

ppRYYGrrYB225

YrR8RrY



例题 1039 到底买不买

- 输入格式：
- 每个输入包含 1 个测试用例。每个测试用例分别在 2 行中先后给出摊主的珠串和小红想做的珠串，两串都不超过 1000 个珠子。
- 输出格式：
- 如果可以买，则在一行中输出“ Yes” 以及有多少多余的珠子；如果不可以买，则在一行中输出“ No” 以及缺了多少珠子。其间以 1 个空格分隔。

例题 1039 到底买不买

- 思路：怎样的情况要买，怎样的情况不要买?

例题 1039 到底买不买

- 思路：怎样的情况要买，怎样的情况不要买?
- B 串中每个字符都能在 A 串中找一个匹配的消掉

例题 1039 到底买不买

- 思路：怎样的情况要买，怎样的情况不要买?
- B 串中每个字符都能在 A 串中找一个匹配的消掉
- B 串中每一种字符， A 串中出现的次数都不少于 B 串中的!

例题 1039 到底买不买

- 思路：怎样的情况要买，怎样的情况不要买?
- B 串中每个字符都能在 A 串中找一个匹配的消掉
- B 串中每一种字符， A 串中出现的次数都不少于 B 串中的!
- 统计每一种字符在 A 、 B 串中分别出现的次数!

例题 1039 到底买不买

- 思路：多了怎么算多多少？少了怎么算少多少？

例题 1039 到底买不买

- 思路：多了怎么算多多少？少了怎么算少多少?
- 如果数量够， A 串比最终匹配的多多少字符就是多多少
- 如果数量不够，最终匹配的比 B 串长度少多少就是少多少

例题 1039 到底买不买

- 思路：最终怎么做!
- 统计 A 、 B 串中每种字符出现次数
- 对每种字符，统计能匹配多少个（即在 A 、 B 串中出现个数的较小值），并对所有种类字符的匹配次数求 sum
- 如果 (sum==B 串长度) ，则买

  多了 (A 串长度 -sum) 个
- 否则不买，少了 (B 串长度 -sum) 个

例题 1039 到底买不买

```cpp
#include <stdio.h>
#include <string.h>
#include <algorithm>
using namespace std;

const int maxn = 1e3 + 5;

char a[maxn],b[maxn];
int numa[maxn],numb[maxn],sum;

int main(){
    scanf("%s%s",a,b);
    for(int i = 0 ; a[i] ; ++i)numa[a[i]]++;
    for(int i = 0 ; b[i] ; ++i)numb[b[i]]++;
    for(int i = 0 ; i < 128 ; ++i)sum += min(numa[i],numb[i]);
    if(sum == strlen(b))printf("Yes %d\n",strlen(a) - sum);
    else printf("No %d\n",strlen(b) - sum);
    return 0;
}
```

例题 1067 试密码

- 当你试图登录某个系统却忘了密码时，系统一般只会允许你尝试有限多次，当超出允许次数时，账号就会被锁死。本题就请你实现这个小功能。
- 输入格式：
- 输入在第一行给出一个密码（长度不超过 20 的、不包含空格、 Tab、回车的非空字符串）和一个正整数 N （<= 10），分别是正确的密码和系统允许尝试的次数。随后每行给出一个以回车结束的非空字符串，是用户尝试输入的密码。输入保证至少有一次尝试。当读到一行只有单个 # 字符时，输入结束，并且这一行不是用户的输入。
- 输出格式：
- 对用户的每个输入，如果是正确的密码且尝试次数不超过 N，则在一行中输出“Welcome in”，并结束程序；如果是错误的，则在一行中按格式输出“Wrong password: 用户输入的错误密码”；当错误尝试达到 N 次时，再输出一行“Account locked”，并结束程序。

例题 1067 试密码

- 输入样例 1：
- Correct%pw 3
- correct%pw
- Correct@PW
- whatisthepassword!
- Correct%pw
- #
- 输出样例 1：
- Wrong password: correct%pw
- Wrong password: Correct@PW
- Wrong password: whatisthepassword!
- Account locked

- 输入样例 2：
- cool@gplt 3
- coolman@gplt
- coollady@gplt
- cool@gplt
- try again
- #
- 输出样例 2：
- Wrong password: coolman@gplt
- Wrong password: coollady@gplt
- Welcome in

例题 1067 试密码

- 思路：首先这题是一道模拟题，思路很明确，就是按题面描述，输出密码正确、密码错误、密码锁定

例题 1067 试密码

- 思路：首先这题是一道模拟题，思路很明确，就是按题面描述，输出密码正确、密码错误、密码锁定
- 注意点：
  - 尝试密码的次数并没有给出，可能尝试了不止 10 次，或者很多
  - 尝试的密码没有说无空格、也没有说长度是多少，建议用 gets 、字符数组长度开大一点

例题 1067 试密码

- 思路：首先这题是一道模拟题，思路很明确，就是按题面描述，输出密码正确、密码错误、密码锁定
- 要点：用计数器记录尝试次数，一旦密码错误且次数已达上限，额外输出密码锁定并结束程序

例题 1067 试密码

- 思路：首先这题是一道模拟题，思路很明确，就是按题面描述，输出密码正确、密码错误、密码锁定
- 问题：那读入呢？要全部读完吗？一定要读到"#\0" 吗？

测评机制

- 测评机到底如何判定你的程序是对的?

测评机制

- 测评机到底如何判定你的程序是对的?
- 在满足时间、空间的限制之内，只要你的输出是对的，你的程序就是对的!
- 除最后一行的行末回车外，对所有字符进行文本匹配，匹配结果完全一致则认为运行通过!
- 测评机并不管你到底读入了多少内容，就算你什么都不读，只要输出结果对，你的程序就是对的!

例题 1067 试密码

- 思路：首先这题是一道模拟题，思路很明确，就是按题面描述，输出密码正确、密码错误、密码锁定
- 问题：那读入呢？要全部读完吗？一定要读到"#\0" 吗？
- 回答：题目中描述的结束程序，可以放心用 return 0;

例题 1067 试密码

```
char s[25],t[10005];

int main(){
    int n;
    scanf("%s%d",s,&n);
    gets(t);
    while(1){
        gets(t);
        if(t[0] == '#' && t[1] == 0)return 0;
        if(strcmp(s,t) == 0){
            printf("Welcome in\n");
            return 0;
        }
        else{
            printf("Wrong password: %s\n",t);
            n--;
            if(n == 0){
                printf("Account locked\n");
                return 0;
            }
        }
    }
    return 0;
}
```

map 的简单使用方法

- count 或这 vis 或者 match 这样的数组，数组下标存放索引，数组的值存放对应索引的值，想必已经并不陌生了

map 的简单使用方法

- count 或这 vis 或者 match 这样的数组，数组下标存放索引，数组的值存放对应索引的值，想必已经并不陌生了
- 以上其实都属于 map 的一种简便实现
- 而 map 则更加灵活，它的数组下标可以是任意可排序的数据类型（ int,long long,string, 甚至是你自己重载过小于号的 struct ）
- 而 map 同样可以与数组一样使用

map 的简单使用方法

```
#include <map>
using namespace std;

map<int,int> M;
map<string,int> M;
map<long long,string> M;
```

map 的简单使用方法

map<int,int> M;
map<string,int> M;
map<long long,string> M;

- 初学者该如何理解?
- map<type1,type2> M; → int a;
- 数据类型 变量名
- 你可以将定义的变量 M 看作一个类似数组的东西
- type1 -> 数组下标的类型
- type2 -> 数组元素的类型

map 的简单使用方法

map<string,int> M;

- 如何使用?
- string a = string(“abc”);
- M[a] = 1;
- M[string(“abc”)] = 1;

map 的简单使用方法

map<string,int> M;

- 如何使用?
- string a = string(“abc”);
- M[a] = 1;
- M[string(“abc”)] = 1;
-
- map 在任何位置定义，其元素值都自动初始化为 0

map 的简单使用方法

map<string,int> M;

- 如何使用?
- string a = string(“abc”);
- M[a] = 1;
- M[string(“abc”)] = 1;
-
- map 在任何位置定义，其元素值都自动初始化为 0
- 在 0 可能被使用时，如何判断元素是否存在?
- M.find(string(“abc”)) == M.end() → 不存在

例题 1069 微博转发抽奖

- 小明 PAT 考了满分，高兴之余决定发起微博转发抽奖活动，从转发的网友中按顺序每隔 N 个人就发出一个红包。请你编写程序帮助他确定中奖名单。
- 输入格式：
- 输入第一行给出三个正整数 M （<= 1000）、 N 和 S，分别是转发的总量、小明决定的中奖间隔、以及第一位中奖者的序号（编号从 1 开始）。随后 M 行，顺序给出转发微博的网友的昵称（不超过 20 个字符、不包含空格回车的非空字符串）。
- 注意：可能有人转发多次，但不能中奖多次。所以如果处于当前中奖位置的网友已经中过奖，则跳过他顺次取下一位。
- 输出格式：
- 按照输入的顺序输出中奖名单，每个昵称占一行。如果没有人中奖，则输出“ Keep going...”。

例题 1069 微博转发抽奖

- 输入样例 1：
- 9 3 2
- Imgonnawin!
- PickMe
- PickMeMeMeee
- LookHere
- Imgonnawin!
- TryAgainAgain
- TryAgainAgain
- Imgonnawin!
- TryAgainAgain
- 输出样例 1：
- PickMe
- Imgonnawin!
- TryAgainAgain

- 输入样例 2：
- 2 3 5
- Imgonnawin!
- PickMe
- 输出样例 2：
- Keep going...

例题 1069 微博转发抽奖

- 思路：
  - 没人中奖
  - 有人中奖
    - 中奖位置的人没拿过奖——得奖
    - 中奖位置的人拿过奖了——中奖位置后移

例题 1069 微博转发抽奖

- 思路：
  - 没人中奖
  - 有人中奖
    - 中奖位置的人没拿过奖——得奖
    - 中奖位置的人拿过奖了——中奖位置后移
- 重点：
  - 怎么样知道一个人是否中过奖
  - 怎么样确定中奖位置以及怎样位置顺延

例题 1069 微博转发抽奖

- 怎么样知道一个人是否中过奖
  - map<string,int> M;
  - M[s] = 0 表示未中过奖
  - M[s] = 1 表示中过奖
  - 中奖时进行赋值 1

例题 1069 微博转发抽奖

- 怎么样确定中奖位置以及怎样位置顺延
  - 首先遍历到第一个获奖位置——必定获奖

    从上一个获奖位置开始，向后 n 个位置为获奖位置
  - 我们在有人获奖时，将计数器 cnt 清零

    之后每经过一个位置， cnt+1

    当 cnt 为 n 时，这个位置应该获奖
  - 顺延？由于获奖位置应该满足 cnt 为 n

    为了使遍历下一个位置时， cnt 又为 n ，我们在需要顺延的时候令 cnt-1

例题 1069 微博转发抽奖

```
for(int i = 1 ; i <= m ; ++i){
    scanf("%s",s);
    string ss(s);
    if(i < stx)continue;
    if(i == stx){
        printf("%s\n",s);
        M[ss] = 1;
        cnt = 0;
    }
    else{
        cnt++;
        if(cnt == n){
            if(M[ss] == 0){
                printf("%s\n",s);
                M[ss] = 1;
                cnt = 0;
            }
            else cnt--;
        }
    }
}
```

谢谢！

也祝各位都能取得满意的成绩～