PAT乙级的题库

[1001 害死人不偿命的(3n+1)猜想 3](#_Toc4451357)

[1002 写出这个数 (20) 4](#_Toc4451358)

[1003 我要通过！(20) 5](#_Toc4451359)

[1004 成绩排名 (20) 7](#_Toc4451360)

[1005 继续(3n+1)猜想 8](#_Toc4451361)

[1006 换个格式输出整数 10](#_Toc4451362)

[1007. 素数对猜想 12](#_Toc4451363)

[1008 数组元素循环右移问题 13](#_Toc4451364)

[1009 说反话 14](#_Toc4451365)

[1010 一元多项式求导 15](#_Toc4451366)

[1011 A+B和C 16](#_Toc4451367)

[1012 数字分类 17](#_Toc4451368)

[1013 数素数 19](#_Toc4451369)

[1014 福尔摩斯的约会 21](#_Toc4451370)

[1015 德才论 23](#_Toc4451371)

[1016 部分A+B 25](#_Toc4451372)

[1017 A除以B 27](#_Toc4451373)

[1018 锤子剪刀布 28](#_Toc4451374)

[1020 月饼 33](#_Toc4451375)

[1021 个位数统计 35](#_Toc4451376)

[1022 D进制的A+B 36](#_Toc4451377)

[1023 组个最小数 37](#_Toc4451378)

[1024 科学计数法 39](#_Toc4451379)

[1025 反转链表 41](#_Toc4451380)

[1026 程序运行时间 45](#_Toc4451381)

[1027 打印沙漏 46](#_Toc4451382)

[1028 人口普查 47](#_Toc4451383)

[1029 旧键盘 49](#_Toc4451384)

[1030 完美数列 50](#_Toc4451385)

[1031 查验身份证 51](#_Toc4451386)

[1032 挖掘机技术哪家强 53](#_Toc4451387)

[1033 旧键盘打字 54](#_Toc4451388)

[1034 有理数四则运算 56](#_Toc4451389)

[1035 插入与归并 58](#_Toc4451390)

[1036 跟奥巴马一起编程 61](#_Toc4451391)

[1037 在霍格沃茨找零钱 62](#_Toc4451392)

[1038 统计同成绩学生 64](#_Toc4451393)

[1039 到底买不买 65](#_Toc4451394)

[1040 有几个PAT 67](#_Toc4451395)

[1041 考试座位号 68](#_Toc4451396)

[1042 字符统计 69](#_Toc4451397)

[1043 输出PATest 70](#_Toc4451398)

[1044 火星数字 71](#_Toc4451399)

[1045 快速排序 74](#_Toc4451400)

[1047. 编程团体赛 76](#_Toc4451401)

[1048 数字加密 77](#_Toc4451402)

[1049 数列的片段和 79](#_Toc4451403)

[1050 螺旋矩阵 80](#_Toc4451404)

[1051 复数乘法 82](#_Toc4451405)

[1052 卖个萌 82](#_Toc4451406)

[1053 住房空置率 84](#_Toc4451407)

[1054 求平均值 85](#_Toc4451408)

[1055 集体照 87](#_Toc4451409)

[1056 组合数的和 89](#_Toc4451410)

[1057 数零壹 90](#_Toc4451411)

[1058 选择题 90](#_Toc4451412)

[1059 C语言竞赛 93](#_Toc4451413)

[1061 判断题 96](#_Toc4451414)

[1062 最简分数 97](#_Toc4451415)

[1063 计算谱半径 98](#_Toc4451416)

[1064 朋友数 99](#_Toc4451417)

[1065 单身狗 101](#_Toc4451418)

[1066 图像过滤 102](#_Toc4451419)

[1067 试密码 103](#_Toc4451420)

[1068 万绿丛中一点红 105](#_Toc4451421)

[1069 微博转发抽奖 107](#_Toc4451422)

[1070 结绳（贪心） 109](#_Toc4451423)

1001 害死人不偿命的(3n+1)猜想 （15 分)

卡拉兹(Callatz)猜想：

对任何一个正整数 *n*，如果它是偶数，那么把它砍掉一半；如果它是奇数，那么把 (3*n*+1) 砍掉一半。这样一直反复砍下去，最后一定在某一步得到 *n*=1。卡拉兹在 1950 年的世界数学家大会上公布了这个猜想，传说当时耶鲁大学师生齐动员，拼命想证明这个貌似很傻很天真的命题，结果闹得学生们无心学业，一心只证 (3*n*+1)，以至于有人说这是一个阴谋，卡拉兹是在蓄意延缓美国数学界教学与科研的进展……

我们今天的题目不是证明卡拉兹猜想，而是对给定的任一不超过 1000 的正整数 *n*，简单地数一下，需要多少步（砍几下）才能得到 *n*=1？

输入格式：

每个测试输入包含 1 个测试用例，即给出正整数 *n* 的值。

输出格式：

输出从 *n* 计算到 1 需要的步数。

输入样例：

3

输出样例：

5

思路：

可以递归也可以循环，递归算最简单

代码：

#include <iostream>

using namespace std;

int \_3n(int n)

{

if(n==1) return 0;

if(n&1) return \_3n((n\*3+1)/2)+1;

else return \_3n(n/2)+1;

}

int main()

{

int n;

cin>>n;

cout<<\_3n(n)<<endl;

return 0;

}

## 1002 写出这个数 (20)

读入一个正整数 *n*，计算其各位数字之和，用汉语拼音写出和的每一位数字。

### 输入格式：

每个测试输入包含 1 个测试用例，即给出自然数 *n* 的值。这里保证 *n* 小于 10​100​​。

### 输出格式：

在一行内输出 *n* 的各位数字之和的每一位，拼音数字间有 1 空格，但一行中最后一个拼音数字后没有空格。

### 输入样例：

1234567890987654321123456789

### 输出样例：

yi san wu

思路：

水题

代码：

#include <bits/stdc++.h>

using namespace std;

string shu[]={"ling","yi","er","san","si","wu","liu","qi","ba","jiu","shi"};

int main()

{

char ch,num[100];

int sum=0;

while(~scanf("%c",&ch)&&ch!='\n')

{

sum+=ch-'0';

}

sprintf(num,"%d",sum);

cout<<shu[num[0]-'0'];

for(int i=1;i<strlen(num);i++)

{

cout<<' '<<shu[num[i]-'0'];

}

cout<<endl;

return 0;

}

## 1003 我要通过！(20)

“**答案正确**”是自动判题系统给出的最令人欢喜的回复。本题属于 PAT 的“**答案正确**”大派送 —— 只要读入的字符串满足下列条件，系统就输出“**答案正确**”，否则输出“**答案错误**”。

得到“**答案正确**”的条件是：

1. 字符串中必须仅有 P、 A、 T这三种字符，不可以包含其它字符；
2. 任意形如 xPATx 的字符串都可以获得“**答案正确**”，其中 x 或者是空字符串，或者是仅由字母 A 组成的字符串；
3. 如果 aPbTc 是正确的，那么 aPbATca 也是正确的，其中 a、 b、 c 均或者是空字符串，或者是仅由字母 A 组成的字符串。

现在就请你为 PAT 写一个自动裁判程序，判定哪些字符串是可以获得“**答案正确**”的。

### 输入格式：

每个测试输入包含 1 个测试用例。第 1 行给出一个正整数 *n* (<10)，是需要检测的字符串个数。接下来每个字符串占一行，字符串长度不超过 100，且不包含空格。

### 输出格式：

每个字符串的检测结果占一行，如果该字符串可以获得“**答案正确**”，则输出 YES，否则输出 NO。

### 输入样例：

8

PAT

PAAT

AAPATAA

AAPAATAAAA

xPATx

PT

Whatever

APAAATAA

### 输出样例：

YES

YES

YES

YES

NO

NO

NO

NO

思路：

字符串题目，一直都是比较弱也是怕的题目比较考验细心和耐心，这个要注意到题目上递推的关系

代码：

#include <bits/stdc++.h>

using namespace std;

int judge\_1(char s[])

{

int l=strlen(s),p=1,t=1;

for(int i=0;i<l;i++)

{

if(!(s[i]=='P'||s[i]=='A'||s[i]=='T'))

return 0;

if(s[i]=='P')

{

if(p) p=0;

else return 0;

}

if(s[i]=='T')

{

if(t) t=0;

else return 0;

}

}

return 1;

}

int judge\_2(char s[])

{

int l=strlen(s),p,t;

for(int i=0;i<l;i++)

{

if(s[i]=='P') p=i;

if(s[i]=='T') t=i;

}

if(t-p>1&&p\*(t-p-1)==l-t-1)

{

return 1;

}

else return 0;

}

int judge(char s[])

{

return (judge\_1(s)&&judge\_2(s));

}

int main()

{

int n;

char s[105];

cin>>n;

while(n--)

{

scanf("%s",s);

if(judge(s))

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

## 1004 成绩排名 (20)

读入 *n*（>0）名学生的姓名、学号、成绩，分别输出成绩最高和成绩最低学生的姓名和学号。

### 输入格式：

每个测试输入包含 1 个测试用例，格式为

第 1 行：正整数 n

第 2 行：第 1 个学生的姓名 学号 成绩

第 3 行：第 2 个学生的姓名 学号 成绩

... ... ...

第 n+1 行：第 n 个学生的姓名 学号 成绩

其中姓名和学号均为不超过 10 个字符的字符串，成绩为 0 到 100 之间的一个整数，这里保证在一组测试用例中没有两个学生的成绩是相同的。

### 输出格式：

对每个测试用例输出 2 行，第 1 行是成绩最高学生的姓名和学号，第 2 行是成绩最低学生的姓名和学号，字符串间有 1 空格。

### 输入样例：

3

Joe Math990112 89

Mike CS991301 100

Mary EE990830 95

### 输出样例：

Mike CS991301

Joe Math990112

思路：

直接比较

代码：

#include <iostream>

using namespace std;

int main()

{

string name\_max,xh\_max,name\_min,xh\_min,s,s2;

int max=-1,min=999,ans,n;

cin>>n;

while(n--)

{

cin>>s>>s2>>ans;

if(ans>max)

{

max=ans;

name\_max=s;

xh\_max=s2;

}

if(ans<min)

{

min=ans;

name\_min=s;

xh\_min=s2;

}

}

cout<<name\_max<<' '<<xh\_max<<endl<<name\_min<<' '<<xh\_min<<endl;

return 0;

}

1005 继续(3n+1)猜想 (25)

卡拉兹(Callatz)猜想已经在1001中给出了描述。在这个题目里，情况稍微有些复杂。

当我们验证卡拉兹猜想的时候，为了避免重复计算，可以记录下递推过程中遇到的每一个数。例如对 *n*=3 进行验证的时候，我们需要计算 3、5、8、4、2、1，则当我们对 *n*=5、8、4、2 进行验证的时候，就可以直接判定卡拉兹猜想的真伪，而不需要重复计算，因为这 4 个数已经在验证3的时候遇到过了，我们称 5、8、4、2 是被 3“覆盖”的数。我们称一个数列中的某个数 *n* 为“关键数”，如果 *n* 不能被数列中的其他数字所覆盖。

现在给定一系列待验证的数字，我们只需要验证其中的几个关键数，就可以不必再重复验证余下的数字。你的任务就是找出这些关键数字，并按从大到小的顺序输出它们。

### 输入格式：

每个测试输入包含 1 个测试用例，第 1 行给出一个正整数 *K* (<100)，第 2 行给出 *K* 个互不相同的待验证的正整数 *n* (1<*n*≤100)的值，数字间用空格隔开。

### 输出格式：

每个测试用例的输出占一行，按从大到小的顺序输出关键数字。数字间用 1 个空格隔开，但一行中最后一个数字后没有空格。

### 输入样例：

6

3 5 6 7 8 11

### 输出样例：

7 6

思路：

暴力

代码：

#include <iostream>

#include <map>

#include <algorithm>

using namespace std;

map<int ,int>mp;

int \_3n(int n)

{

if(n==1) return 1;

if(n&1)

{

if(!mp[(n\*3+1)/2]) mp[(n\*3+1)/2]=\_3n((n\*3+1)/2);

return mp[(n\*3+1)/2]+1;

}

else

{

if(!mp[n/2]) mp[n/2]=\_3n(n/2);

return mp[n/2]+1;

}

}

int main()

{

int n,a[105],flag=1;

cin>>n;

for(int i=0;i<n;i++)

{

cin>>a[i];

mp[a[i]]=0;

}

sort(a,a+n);

for(int i=n-1;i>=0;i--)

{

\_3n(a[i]);

}

for(map<int,int>::reverse\_iterator it=mp.rbegin();it!=mp.rend();it++)

{

//cout<<"key:"<<it->first<<' '<<"ve:"<<it->second<<endl;

if(it->second==0)

{

if(flag)

{

cout<<it->first;

flag=0;

}

else

{

cout<<' '<<it->first;

}

}

}

cout<<endl;

return 0;

}

1006 换个格式输出整数 (15)

让我们用字母 B 来表示“百”、字母 S 表示“十”，用 12...n 来表示不为零的个位数字 n（<10），换个格式来输出任一个不超过 3 位的正整数。例如 234 应该被输出为 BBSSS1234，因为它有 2 个“百”、3 个“十”、以及个位的 4。

**输入格式：**

每个测试输入包含 1 个测试用例，给出正整数 *n*（<1000）。

### 输出格式：

每个测试用例的输出占一行，用规定的格式输出 *n*。

### 输入样例 1：

234

### 输出样例 1：

BBSSS1234

### 输入样例 2：

23

### 输出样例 2：

SS123

思路：

水

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

char s[5];

cin>>n;

sprintf(s,"%03d",n);

for(int i=0;i<s[0]-'0';i++) printf("B");

for(int i=0;i<s[1]-'0';i++) printf("S");

for(int i=0;i<s[2]-'0';i++) printf("%d",i+1);

cout<<endl;

return 0;

}

1007. 素数对猜想 (20)

让我们定义*d*​*n*​​为：*d*​*n*​​=*p*​*n*+1​​−*p*​*n*​​，其中*p*​*i*​​是第*i*个素数。显然有*d*​1​​=1，且对于*n*>1有*d*​*n*​​是偶数。“素数对猜想”认为“存在无穷多对相邻且差为2的素数”。

现给定任意正整数N(<10​5​​)，请计算不超过N的满足猜想的素数对的个数。

### 输入格式:

输入在一行给出正整数N。

### 输出格式:

在一行中输出不超过N的满足猜想的素数对的个数。

### 输入样例:

20

### 输出样例:

4

思路：

水

代码：

#include <bits/stdc++.h>

using namespace std;

int judeg(int n)

{

for(int i=2;i<sqrt(n)+1;i++)

{

if(n%i==0) return 0;

}

return 1;

}

void init(int a[],int n)

{

int k=0;

for(int i=2;i<=n+1;i++)

{

if(judeg(i)) a[k++]=i;

}

}

int main()

{

int n,a[10005],sum=0;

cin>>n;

init(a,n);

for(int i=0;a[i+1]<=n;i++)

{

if(a[i+1]-a[i]==2) sum++;

}

cout<<sum<<endl;

return 0;

}

1008 数组元素循环右移问题 (20)

一个数组*A*中存有*N*（>0）个整数，在不允许使用另外数组的前提下，将每个整数循环向右移*M*（≥0）个位置，即将*A*中的数据由（*A*​0​​*A*​1​​⋯*A*​*N*−1​​）变换为（*A*​*N*−*M*​​⋯*A*​*N*−1​​*A*​0​​*A*​1​​⋯*A*​*N*−*M*−1​​）（最后*M*个数循环移至最前面的*M*个位置）。如果需要考虑程序移动数据的次数尽量少，要如何设计移动的方法？

### 输入格式:

每个输入包含一个测试用例，第1行输入*N*（1≤*N*≤100）和*M*（≥0）；第2行输入*N*个整数，之间用空格分隔。

### 输出格式:

在一行中输出循环右移*M*位以后的整数序列，之间用空格分隔，序列结尾不能有多余空格。

### 输入样例:

6 2

1 2 3 4 5 6

### 输出样例:

5 6 1 2 3 4

思路：

循环输出，不用储存

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,k,a[105];

cin>>n>>k;

for(int i=0;i<n;i++) cin>>a[i];

for(int i=n-(k%n);i<2\*n-(k%n);i++)

{

printf("%d%c",a[i%n],i==2\*n-(k%n)-1?'\n':' ');

}

return 0;

}

1009 说反话 (20)

给定一句英语，要求你编写程序，将句中所有单词的顺序颠倒输出。

### 输入格式：

测试输入包含一个测试用例，在一行内给出总长度不超过 80 的字符串。字符串由若干单词和若干空格组成，其中单词是由英文字母（大小写有区分）组成的字符串，单词之间用 1 个空格分开，输入保证句子末尾没有多余的空格。

### 输出格式：

每个测试用例的输出占一行，输出倒序后的句子。

### 输入样例：

Hello World Here I Come

### 输出样例：

Come I Here World Hello

思路：

还是水，用一个string的栈存然后输出

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

stack<string>st;

string s;

while(cin>>s) st.push(s);

cout<<st.top();st.pop();

while(!st.empty())

{

cout<<' '<<st.top();

st.pop();

}

cout<<endl;

return 0;

}

1010 一元多项式求导 (25)

设计函数求一元多项式的导数。（注：*x*​*n*​​（*n*为整数）的一阶导数为*nx*​*n*−1​​。）

### 输入格式:

以指数递降方式输入多项式非零项系数和指数（绝对值均为不超过 1000 的整数）。数字间以空格分隔。

### 输出格式:

以与输入相同的格式输出导数多项式非零项的系数和指数。数字间以空格分隔，但结尾不能有多余空格。注意“零多项式”的指数和系数都是 0，但是表示为 0

0。

### 输入样例:

3 4 -5 2 6 1 -2 0

### 输出样例:

12 3 -10 1 6 0

思路：

比较坑，题目上说的不清晰，输入的系数和质数都为0时什么都不输出，如果输入是空输出“0 0”。

代码：

#include <bits/stdc++.h>

using namespace std;

int flag=0;

void fx(int a,int b)

{

if(a\*b)

{

if(flag) cout<<' '; flag=1;

printf("%d %d",a\*b,b-1);

}

}

int main()

{

int a,b;

while(cin>>a>>b)

{

fx(a,b);

}

if(!flag) printf("0 0\n");

return 0;

}

1011 A+B和C (15)

给定区间 [−2​31​​,2​31​​] 内的 3 个整数 *A*、*B* 和 *C*，请判断 *A*+*B* 是否大于 *C*。

### 输入格式：

输入第 1 行给出正整数 *T* (≤10)，是测试用例的个数。随后给出 *T* 组测试用例，每组占一行，顺序给出 *A*、*B* 和 *C*。整数间以空格分隔。

### 输出格式：

对每组测试用例，在一行中输出 Case #X: true 如果 *A*+*B*>*C*，否则输出 Case #X: false，其中 X 是测试用例的编号（从 1 开始）。

### 输入样例：

4

1 2 3

2 3 4

2147483647 0 2147483646

0 -2147483648 -2147483647

### 输出样例：

Case #1: false

Case #2: true

Case #3: true

Case #4: false

思路：

水题不用思路

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

long long a,b,c;

int n;cin>>n;

for(int i=1;i<=n;i++)

{

scanf("%lld%lld%lld",&a,&b,&c);

if((a+b)>c)

printf("Case #%d: true\n",i);

else

printf("Case #%d: false\n",i);

}

return 0;

}

1012 数字分类 (20)

给定一系列正整数，请按要求对数字进行分类，并输出以下 5 个数字：

* *A*​1​​ = 能被 5 整除的数字中所有偶数的和；
* *A*​2​​ = 将被 5 除后余 1 的数字按给出顺序进行交错求和，即计算 *n*​1​​−*n*​2​​+*n*​3​​−*n*​4​​⋯；
* *A*​3​​ = 被 5 除后余 2 的数字的个数；
* *A*​4​​ = 被 5 除后余 3 的数字的平均数，精确到小数点后 1 位；
* *A*​5​​ = 被 5 除后余 4 的数字中最大数字。

输入格式：

每个输入包含 1 个测试用例。每个测试用例先给出一个不超过 1000 的正整数 *N*，随后给出 *N* 个不超过 1000 的待分类的正整数。数字间以空格分隔。

输出格式：

对给定的 *N* 个正整数，按题目要求计算 *A*​1​​~*A*​5​​ 并在一行中顺序输出。数字间以空格分隔，但行末不得有多余空格。

若其中某一类数字不存在，则在相应位置输出 N。

输入样例 1：

13 1 2 3 4 5 6 7 8 9 10 20 16 18

输出样例 1：

30 11 2 9.7 9

输入样例 2：

8 1 2 4 5 6 7 9 16

输出样例 2：

N 11 2 N 9

思路：

按照要求分类就好，比较繁琐，写了无数if -\_-!

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;cin>>n;

int a1=0,s1=0,a2=0,s2=0,a3=0,s3=0,a4=0,s4=0,a5=-0x3f3f3f3f,s5=0;

while(n--)

{

int ans;

cin>>ans;

switch(ans%5)

{

case 0:

if(!(ans%2))

{

a1+=ans;

s1++;

}

break;

case 1:

if(!(s2%2))

a2+=ans;

else

a2-=ans;

s2++;

break;

case 2:

s3++;

break;

case 3:

a4+=ans;

s4++;

break;

case 4:

a5=max(a5,ans);

s5++;

break;

}

}

if(!s1)

cout<<'N'<<' ';

else

cout<<a1<<' ';

if(!s2)

cout<<'N'<<' ';

else

cout<<a2<<' ';

if(!s3)

cout<<'N'<<' ';

else

cout<<s3<<' ';

if(!s4)

cout<<'N'<<' ';

else

printf("%.1lf ",a4\*1.0/s4);

if(!s5)

cout<<'N'<<endl;

else

cout<<a5<<endl;

return 0;

}

1013 数素数 (20)

令 *P*​*i*​​ 表示第 *i* 个素数。现任给两个正整数 *M*≤*N*≤10​4​​，请输出 *P*​*M*​​ 到 *P*​*N*​​ 的所有素数。

### 输入格式：

输入在一行中给出 *M* 和 *N*，其间以空格分隔。

### 输出格式：

输出从 *P*​*M*​​ 到 *P*​*N*​​ 的所有素数，每 10 个数字占 1 行，其间以空格分隔，但行末不得有多余空格。

### 输入样例：

5 27

### 输出样例：

11 13 17 19 23 29 31 37 41 43

47 53 59 61 67 71 73 79 83 89

97 101 103

思路：

打表存1w个素数，筛法打表，普通方法不知道超不超，然后输出就好,隔是个要换行一次。

代码：

#include <bits/stdc++.h>

using namespace std;

int a[104799]={0},su[10005];

void init()

{

int k=0;

a[0]=a[1]=1;

for(int i=2;i<104799&&k<10005;i++)

{

if(!a[i])

{

su[k]=i;

k++;

for(int j=i+i;j<104799;j+=i)

{

a[j]=1;

}

}

}

}

int main()

{

init();

int n,m;

cin>>n>>m;

for(int i=n;i<=m;i++)

{

if((i-n+1)%10==0||i==m)

printf("%d\n",su[i-1]);

else

printf("%d ",su[i-1]);

}

return 0;

}

1014 福尔摩斯的约会 (20)

大侦探福尔摩斯接到一张奇怪的字条：我们约会吧！ 3485djDkxh4hhGE 2984akDfkkkkggEdsb s&hgsfdk d&Hyscvnm。大侦探很快就明白了，字条上奇怪的乱码实际上就是约会的时间星期四 14:04，因为前面两字符串中第 1 对相同的大写英文字母（大小写有区分）是第 4 个字母 D，代表星期四；第 2 对相同的字符是 E ，那是第 5 个英文字母，代表一天里的第 14 个钟头（于是一天的 0 点到 23 点由数字 0 到 9、以及大写字母 A 到 N 表示）；后面两字符串第 1 对相同的英文字母 s 出现在第 4 个位置（从 0 开始计数）上，代表第 4 分钟。现给定两对字符串，请帮助福尔摩斯解码得到约会的时间。

### 输入格式：

输入在 4 行中分别给出 4 个非空、不包含空格、且长度不超过 60 的字符串。

### 输出格式：

在一行中输出约会的时间，格式为 DAY HH:MM，其中 DAY 是某星期的 3 字符缩写，即 MON 表示星期一，TUE 表示星期二，WED 表示星期三，THU 表示星期四，FRI 表示星期五，SAT 表示星期六，SUN 表示星期日。题目输入保证每个测试存在唯一解。

### 输入样例：

3485djDkxh4hhGE

2984akDfkkkkggEdsb

s&hgsfdk

d&Hyscvnm

### 输出样例：

THU 14:04

思路：

这个题略坑，刚开始理解错了两次都以为是顺序往下找相同的一对，结果只用两行上下对比是不是相同就行了。然后就是注意题目上的什么大写字母，字母，数字分清每一个要找那种相同的就行了

代码：

#include <bits/stdc++.h>

using namespace std;

string DAY[7]={"MON","TUE","TUE","THU","FRI","SAT","SUN"};

int main()

{

char s[200],s2[100];

scanf("%s",s);

int l=strlen(s),l2,flag=0,hh,mm,day;

scanf("%s",s2);

l2=strlen(s2);

map<char, int>mp;

for(int i=0;i<l;i++)

mp[s[i]]=1;

for(int i=0;i<l2;i++)

{

if(s2[i]==s[i])

{

if(!flag&&s2[i]>='A'&&s2[i]<='Z')

{

flag=1;

day=s2[i]-'A';

}

else if(flag&&(s2[i]>='A'&&s2[i]<='N'||s2[i]>='0'&&s2[i]<='9'))

{

if(s[i]>='0'&&s[i]<='9')

hh = s2[i]-'0';

else

hh = s2[i]-'A'+10;

break;

}

}

}

scanf("%s",s);

l=strlen(s);

scanf("%s",s2);

l2=strlen(s2);

mp.clear();

for(int i=0;i<l;i++)

mp[s[i]]=1;

for(int i=0;i<l2;i++)

{

if(s2[i]==s[i]&&(s2[i]>='A'&&s2[i]<='Z'||s2[i]>='a'&&s2[i]<='z'))

{

mm=i;

break;

}

}

cout<<DAY[day];

printf(" %02d:%02d\n",hh,mm);

return 0;

}

1015 德才论 (25)

宋代史学家司马光在《资治通鉴》中有一段著名的“德才论”：“是故才德全尽谓之圣人，才德兼亡谓之愚人，德胜才谓之君子，才胜德谓之小人。凡取人之术，苟不得圣人，君子而与之，与其得小人，不若得愚人。”

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：

输入第一行给出 3 个正整数，分别为：*N*（≤10​5​​），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。

### 输出格式：

输出第一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。

### 输入样例：

14 60 80

10000001 64 90

10000002 90 60

10000011 85 80

10000003 85 80

10000004 80 85

10000005 82 77

10000006 83 76

10000007 90 78

10000008 75 79

10000009 59 90

10000010 88 45

10000012 80 100

10000013 90 99

10000014 66 60

### 输出样例：

12

10000013 90 99

10000012 80 100

10000003 85 80

10000011 85 80

10000004 80 85

10000007 90 78

10000006 83 76

10000005 82 77

10000002 90 60

10000014 66 60

10000008 75 79

10000001 64 90

思路：

排序的题目，不过条件太多并且麻烦需要细心的分类和写cmp函数，定义一个结构体储存一个人的所有信息，只要分类和cmp函数没搞错，就直接用sort排序输出就行了。

代码：

#include <bits/stdc++.h>

using namespace std;

struct node{

int id,de,ca,lei,zong;

};

int cmp(node a,node b)

{

if(a.lei==b.lei)

{

if(a.zong==b.zong)

{

if(a.de==b.de)

return a.id<b.id;

return a.de>b.de;

}

return a.zong>b.zong;

}

return a.lei<b.lei;

}

int main()

{

int sum=0,n,l,h;

node a[100005],ans;

cin>>n>>l>>h;

while(n--)

{

scanf("%d %d %d",&ans.id,&ans.de,&ans.ca);

if(ans.ca>=l&&ans.de>=l)

{

ans.zong=ans.ca+ans.de;

if(ans.de>=h)

{

if(ans.ca>=h)

ans.lei=1;

else

ans.lei=2;

}

else

{

if(ans.de>=ans.ca)

ans.lei=3;

else

ans.lei=4;

}

a[sum++]=ans;

}

}

sort(a,a+sum,cmp);

cout<<sum<<endl;

for(int i=0;i<sum;i++)

{

printf("%d %d %d\n",a[i].id,a[i].de,a[i].ca);

}

return 0;

}

1016 部分A+B (15)

正整数 *A* 的“*D*​*A*​​（为 1 位整数）部分”定义为由 *A* 中所有 *D*​*A*​​ 组成的新整数 *P*​*A*​​。例如：给定 *A*=3862767，*D*​*A*​​=6，则 *A* 的“6 部分”*P*​*A*​​ 是 66，因为 *A*中有 2 个 6。

现给定 *A*、*D*​*A*​​、*B*、*D*​*B*​​，请编写程序计算 *P*​*A*​​+*P*​*B*​​。

### 输入格式：

输入在一行中依次给出 *A*、*D*​*A*​​、*B*、*D*​*B*​​，中间以空格分隔，其中 0<*A*,*B*<10​10​​。

### 输出格式：

在一行中输出 *P*​*A*​​+*P*​*B*​​ 的值。

### 输入样例 1：

3862767 6 13530293 3

### 输出样例 1：

399

### 输入样例 2：

3862767 1 13530293 8

### 输出样例 2：

0

思路：

意思很简单，只需要找出Da，Db的数量，然后写一个函数能把pa pb算出来直接相加，找Da的时候可以直接写不麻烦，也可以调用STL中的函数count()（又发现一个函数 ヽ(✿ﾟ▽ﾟ)ノ）。

代码：

#include <bits/stdc++.h>

using namespace std;

int px(int dx,int n)

{

if(n==0) return 0;

if(n==1) return dx;

return px(dx,n-1)\*10+dx;

}

int main()

{

char s[100],s2[100],Da,Db;

scanf("%s %c %s %c",s,&Da,s2,&Db);

printf("%d\n",px(Da-'0',count(s,s+strlen(s),Da))+px(Db-'0',count(s2,s2+strlen(s2),Db)));

return 0;

}

1017 A除以B (20)

本题要求计算 *A*/*B*，其中 *A* 是不超过 1000 位的正整数，*B* 是 1 位正整数。你需要输出商数 *Q* 和余数 *R*，使得 *A*=*B*×*Q*+*R* 成立。

### 输入格式：

输入在一行中依次给出 *A* 和 *B*，中间以 1 空格分隔。

### 输出格式：

在一行中依次输出 *Q* 和 *R*，中间以 1 空格分隔。

### 输入样例：

123456789050987654321 7

### 输出样例：

17636684150141093474 3

思路：

大数问题，不过是大数除10以下整数比较容易，模拟手工算，一位一位的算然后存下来就好了。

牛客过了官网没过，最后发现没有考虑啊a

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

char a[1005],q[1005];

int b,r,ans,j=0,i=0;

scanf("%s %d",a,&b);

int l=strlen(a);

ans=a[0]-'0';i=1;

do

{

if(ans<b&&i<l)

{

ans=ans\*10+a[i++]-'0';

}

q[j++]=ans/b;

ans%=b;

}while(i<l);

for(int i=0;i<j;i++)

printf("%c",q[i]+'0');

printf(" %d\n",ans);

return 0;

}

1018 锤子剪刀布 (20)

大家应该都会玩“锤子剪刀布”的游戏：两人同时给出手势，胜负规则如图所示：



现给出两人的交锋记录，请统计双方的胜、平、负次数，并且给出双方分别出什么手势的胜算最大。

### 输入格式：

输入第 1 行给出正整数 *N*（≤10​5​​），即双方交锋的次数。随后 *N* 行，每行给出一次交锋的信息，即甲、乙双方同时给出的的手势。C 代表“锤子”、J 代表“剪刀”、B 代表“布”，第 1 个字母代表甲方，第 2 个代表乙方，中间有 1 个空格。

### 输出格式：

输出第 1、2 行分别给出甲、乙的胜、平、负次数，数字间以 1 个空格分隔。第 3 行给出两个字母，分别代表甲、乙获胜次数最多的手势，中间有 1 个空格。如果解不唯一，则输出按字母序最小的解。

### 输入样例：

10

C J

J B

C B

B B

B C

C C

C B

J B

B C

J J

### 输出样例：

5 3 2

2 3 5

B B

思路：

按照题意模拟即可（又是一堆if，是我没有get到乙级题的真谛吗 (ಥ \_ ಥ)）

代码：

#include <bits/stdc++.h>

using namespace std;

struct node

{

int sheng,fu,ping;

map<char,int>mp;

node()

{

sheng=fu=ping=0;

}

};

void input\_jcb(node \*J)

{

if(J->mp['B']>=J->mp['C']&&J->mp['B']>=J->mp['J'])

{

printf("B");

}

else if(J->mp['C']>=J->mp['B']&&J->mp['C']>=J->mp['J'])

{

printf("C");

}

else if(J->mp['J']>=J->mp['B']&&J->mp['J']>=J->mp['C'])

{

printf("J");

}

}

int main()

{

int n;

cin>>n;

char ch,ch2;

node \*J = new node;

node \*Y = new node;

while(n--)

{

cin>>ch>>ch2;

if(ch=='B'&&ch2=='B'||ch=='C'&&ch2=='C'||ch=='J'&&ch2=='J')

{

J->ping++;

Y->ping++;

}

if(ch=='B'&&ch2=='C')

{

J->sheng++;

Y->fu++;

J->mp['B']++;

}

if(ch=='B'&&ch2=='J')

{

Y->sheng++;

J->fu++;

Y->mp['J']++;

}

if(ch=='C'&&ch2=='J')

{

J->sheng++;

Y->fu++;

J->mp['C']++;

}

if(ch=='C'&&ch2=='B')

{

Y->sheng++;

J->fu++;

Y->mp['B']++;

}

if(ch=='J'&&ch2=='C')

{

Y->sheng++;

J->fu++;

Y->mp['C']++;

}

if(ch=='J'&&ch2=='B')

{

J->sheng++;

Y->fu++;

J->mp['J']++;

}

}

printf("%d %d %d\n",J->sheng,J->ping,J->fu);

printf("%d %d %d\n",Y->sheng,Y->ping,Y->fu);

input\_jcb(J);

cout<<' ';

input\_jcb(Y);

cout<<endl;

return 0;

}

##大佬代码果然是 短小快！

#include <iostream>

using namespace std;

int win[3] = {0};

char max3(int \*a) {

char r = 'B';

if(a[1] > a[0]) {

a[0] = a[1];

r = 'C';

}

if(a[2] > a[0]) r = 'J';

return r;

}

int main() {

int n;

char a[105],b[105];

int awin[3] = {0};

int bwin[3] = {0};

cin >> n;

for(int i = 0; i < n; i++) {

cin >> a[i] >> b[i];

int k = (a[i] - b[i]) \* (a[i] - b[i]);

if(k == 1)

a[i] == 66 ? win[0]++,awin[0]++ : (win[1]++,bwin[0]++);

else if(k == 49)

a[i] == 67 ? win[0]++,awin[1]++ : (win[1]++,bwin[1]++);

else if(k == 64)

a[i] == 74 ? win[0]++,awin[2]++ : (win[1]++,bwin[2]++);

else

win[2]++;

}

cout << win[0] << " " << win[2] << " " << win[1] << endl;

cout << win[1] << " " << win[2] << " " << win[0] << endl;

cout << max3(awin) << " " << max3(bwin);

return 0;

}

1019 数字黑洞 (20)

给定任一个各位数字不完全相同的 4 位正整数，如果我们先把 4 个数字按非递增排序，再按非递减排序，然后用第 1 个数字减第 2 个数字，将得到一个新的数字。一直重复这样做，我们很快会停在有“数字黑洞”之称的 6174，这个神奇的数字也叫 Kaprekar 常数。

例如，我们从6767开始，将得到

7766 - 6677 = 1089

9810 - 0189 = 9621

9621 - 1269 = 8352

8532 - 2358 = 6174

7641 - 1467 = 6174

... ...

现给定任意 4 位正整数，请编写程序演示到达黑洞的过程。

### 输入格式：

输入给出一个 (0,10​4​​) 区间内的正整数 *N*。

### 输出格式：

如果 *N* 的 4 位数字全相等，则在一行内输出 N - N = 0000；否则将计算的每一步在一行内输出，直到 6174 作为差出现，输出格式见样例。注意每个数字按 4 位数格式输出。

### 输入样例 1：

6767

### 输出样例 1：

7766 - 6677 = 1089

9810 - 0189 = 9621

9621 - 1269 = 8352

8532 - 2358 = 6174

### 输入样例 2：

2222

### 输出样例 2：

2222 - 2222 = 0000

思路：

经典题，字符串与整型相互转换就好了，可以自己写也可以直接用sscanf，sprintf

代码：

#include <bits/stdc++.h>

using namespace std;

int cmp(char a,char b)

{

return a>b;

}

void zhuanhuan(char s[],int \*a,int \*b)

{

sort(s,s+4,cmp);

sscanf(s,"%d",a);

sort(s,s+4);

sscanf(s,"%d",b);

}

int main()

{

int a=0,b=0,n;

char s[10];

cin>>n;

sprintf(s,"%04d",n);

zhuanhuan(s,&a,&b);

while(1)

{

printf("%04d - %04d = %04d\n",a,b,a-b);

if(a-b==6174||a==b)

break;

sprintf(s,"%04d",a-b);

zhuanhuan(s,&a,&b);

}

return 0;

}

1020 月饼 (25)

月饼是中国人在中秋佳节时吃的一种传统食品，不同地区有许多不同风味的月饼。现给定所有种类月饼的库存量、总售价、以及市场的最大需求量，请你计算可以获得的最大收益是多少。

注意：销售时允许取出一部分库存。样例给出的情形是这样的：假如我们有 3 种月饼，其库存量分别为 18、15、10 万吨，总售价分别为 75、72、45 亿元。如果市场的最大需求量只有 20 万吨，那么我们最大收益策略应该是卖出全部 15 万吨第 2 种月饼、以及 5 万吨第 3 种月饼，获得 72 + 45/2 = 94.5（亿元）。

### 输入格式：

每个输入包含一个测试用例。每个测试用例先给出一个不超过 1000 的正整数 *N* 表示月饼的种类数、以及不超过 500（以万吨为单位）的正整数 *D* 表示市场最大需求量。随后一行给出 *N* 个正数表示每种月饼的库存量（以万吨为单位）；最后一行给出 *N* 个正数表示每种月饼的总售价（以亿元为单位）。数字间以空格分隔。

### 输出格式：

对每组测试用例，在一行中输出最大收益，以亿元为单位并精确到小数点后 2 位。

### 输入样例：

3 20

18 15 10

75 72 45

### 输出样例：

94.50

思路：

基础贪心，选择性价比最高的

代码：

#include <bits/stdc++.h>

using namespace std;

struct node

{

double jia,liang,bizhi;

};

int cmp(node a,node b)

{

return a.bizhi>b.bizhi;

}

int main()

{

node a[1005];

int n;

double d,sum=0;

cin>>n>>d;

for(int i=0;i<n;i++)

{

cin>>a[i].liang;

}

for(int i=0;i<n;i++)

{

cin>>a[i].jia;

a[i].bizhi=a[i].jia/a[i].liang;

}

sort(a,a+n,cmp);

for(int i=0;d;i++)

{

if(d>a[i].liang)

{

d-=a[i].liang;

sum+=a[i].jia;

}

else

{

sum+=d\*a[i].bizhi;

d=0;

}

}

printf("%.2lf\n",sum);

return 0;

}

1021 个位数统计 (15)

给定一个 *k* 位整数 *N*=*d*​*k*−1​​10​*k*−1​​+⋯+*d*​1​​10​1​​+*d*​0​​ (0≤*d*​*i*​​≤9, *i*=0,⋯,*k*−1, *d*​*k*−1​​>0)，请编写程序统计每种不同的个位数字出现的次数。例如：给定 *N*=100311，则有 2 个 0，3 个 1，和 1 个 3。

### 输入格式：

每个输入包含 1 个测试用例，即一个不超过 1000 位的正整数 *N*。

### 输出格式：

对 *N* 中每一种不同的个位数字，以 D:M 的格式在一行中输出该位数字 D 及其在 *N* 中出现的次数 M。要求按 D 的升序输出。

### 输入样例：

100311

### 输出样例：

0:2

1:3

3:1

思路：

水题，开一个大小10的数组存每一个的次数，数字就是下标

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int mp[10]={0};

char ch;

while(~scanf("%c",&ch))

{

mp[ch-'0']++;

}

for(int i=0;i<10;i++)

{

if(mp[i])

printf("%d:%d\n",i,mp[i]);

}

return 0;

}

1022 D进制的A+B (20)

输入两个非负 10 进制整数 *A* 和 *B* (≤2​30​​−1)，输出 *A*+*B* 的 *D* (1<*D*≤10)进制数。

### 输入格式：

输入在一行中依次给出 3 个整数 *A*、*B* 和 *D*。

### 输出格式：

输出 *A*+*B* 的 *D* 进制数。

### 输入样例：

123 456 8

### 输出样例：

1103

思路：

就是考察进制转换

牛客过了官网没过，没有考虑0的情况

代码：

#include <bits/stdc++.h>

using namespace std;

void my\_itoa(int n,int r)

{

if(n==0)printf("0");

stack<int>st;

while(n)

{

st.push(n%r);

n/=r;

}

while(!st.empty())

{

cout<<st.top();

st.pop();

}

}

int main()

{

int a,b,d;

cin>>a>>b>>d;

my\_itoa(a+b,d);

return 0;

}

1023 组个最小数 (20)

给定数字 0-9 各若干个。你可以以任意顺序排列这些数字，但必须全部使用。目标是使得最后得到的数尽可能小（注意 0 不能做首位）。例如：给定两个 0，两个 1，三个 5，一个 8，我们得到的最小的数就是 10015558。

现给定数字，请编写程序输出能够组成的最小的数。

### 输入格式：

输入在一行中给出 10 个非负整数，顺序表示我们拥有数字 0、数字 1、……数字 9 的个数。整数间用一个空格分隔。10 个数字的总个数不超过 50，且至少拥有 1 个非 0 的数字。

### 输出格式：

在一行中输出能够组成的最小的数。

### 输入样例：

2 2 0 0 0 3 0 0 1 0

### 输出样例：

10015558

思路：

先找出一个非零的最小数字输出，然后将剩下的数字升序输出就好了

代码：

#include <iostream>

using namespace std;

int main()

{

int mp[10]={0};

for(int i=0;i<10;i++)

{

cin>>mp[i];

}

for(int i=1;i<10;i++)

{

if(mp[i])

{

mp[i]--;

cout<<i;

break;

}

}

for(int i=0;i<10;i++)

{

if(mp[i])

{

for(int j=0;j<mp[i];j++)

cout<<i;

}

}

cout<<endl;

return 0;

}

1024 科学计数法 (20)

科学计数法是科学家用来表示很大或很小的数字的一种方便的方法，其满足正则表达式 [+-][1-9].[0-9]+E[+-][0-9]+，即数字的整数部分只有 1 位，小数部分至少有 1 位，该数字及其指数部分的正负号即使对正数也必定明确给出。

现以科学计数法的格式给出实数 *A*，请编写程序按普通数字表示法输出 *A*，并保证所有有效位都被保留。

### 输入格式：

每个输入包含 1 个测试用例，即一个以科学计数法表示的实数 *A*。该数字的存储长度不超过 9999 字节，且其指数的绝对值不超过 9999。

### 输出格式：

对每个测试用例，在一行中按普通数字表示法输出 *A*，并保证所有有效位都被保留，包括末尾的 0。

### 输入样例 1：

+1.23400E-03

### 输出样例 1：

0.00123400

### 输入样例 2：

-1.2E+10

### 输出样例 2：

-12000000000

思路：

看似复杂，只要明白了题目上科学计数法的规则，实际上还是对字符串的处理

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

char s[10005],ch;

queue<char>qu;

int i=0,flag=1;

while(~scanf("%c",&ch))

{

if(ch=='-')cout<<'-';

else if(ch=='+'||ch=='.');

else if(ch>='0'&&ch<='9')

{

qu.push(ch);

}

else

{

break;

}

}

int ans;

cin>>ans;

flag+=ans;

if(flag<1)

{

for(int i=0; i<=-flag; i++)

{

if(i==1)cout<<'.';

cout<<'0';

}

while(!qu.empty())

{

cout<<qu.front();

qu.pop();

}

}

else

{

while(!qu.empty())

{

if(i==flag)cout<<'.';

cout<<qu.front();

qu.pop();

i++;

}

for(; i<flag; i++)

{

cout<<'0';

}

}

cout<<endl;

return 0;

}

1025 反转链表 (25)

给定一个常数 *K* 以及一个单链表 *L*，请编写程序将 *L* 中每 *K* 个结点反转。例如：给定 *L* 为 1→2→3→4→5→6，*K* 为 3，则输出应该为 3→2→1→6→5→4；如果 *K*为 4，则输出应该为 4→3→2→1→5→6，即最后不到 *K* 个元素不反转。

### 输入格式：

每个输入包含 1 个测试用例。每个测试用例第 1 行给出第 1 个结点的地址、结点总个数正整数 *N* (≤10​5​​)、以及正整数 *K* (≤*N*)，即要求反转的子链结点的个数。结点的地址是 5 位非负整数，NULL 地址用 −1 表示。

接下来有 *N* 行，每行格式为：

Address Data Next

其中 Address 是结点地址，Data 是该结点保存的整数数据，Next 是下一结点的地址。

### 输出格式：

对每个测试用例，顺序输出反转后的链表，其上每个结点占一行，格式与输入相同。

### 输入样例：

00100 6 4

00000 4 99999

00100 1 12309

68237 6 -1

33218 3 00000

99999 5 68237

12309 2 33218

### 输出样例：

00000 4 33218

33218 3 12309

12309 2 00100

00100 1 99999

99999 5 68237

68237 6 -1

思路：

这道题目比较坑卡了我很长时间，刚开始完全就是模拟一个链表出来然后改变指针，写着巨麻烦还wa掉了，后来想到了完全必须要在原来链表上面改动，把新的顺序重新存一遍就好了，反向部分用栈来操作反向，然后用结构体vector来存。写完过来但是也挺长的，看来大佬代码恍然大悟，只用存“指针”部分就行了，然后用c++提供的反向库函数来操作，代码果断比我的短一半。

有一个坑点，输入的节点不一定全用上会有部分不出现在链表中的废节点，所以需要重新计算N的数量

牛客过了官网没过，原因没考虑全部反转和反转后剩一个的情况

代码：

#include <bits/stdc++.h>

using namespace std;

struct node

{

int data,next;

} mp[100005];

struct node2

{

int id,data,next;

};

int main()

{

int head,n,k,ans,flag=1;

vector<node2>ve;

stack<node2>st;

cin>>head>>n>>k;

for(int i=0; i<n; i++)//储存节点

{

scanf("%d",&ans);

scanf("%d %d",&mp[ans].data,&mp[ans].next);

}

ans=head;

n=1;

while(1)

{

if(mp[ans].next!=-1)//遍历链表统计N

{

ans=mp[ans].next;

n++;

}

else

{

break;

}

}

do

{

for(int i=0; i<k; i++)

{

node2 p;

p.data=mp[head].data;

p.id=head;

p.next=mp[head].next;

st.push(p);//存入栈中

head=mp[head].next;

}

while(!st.empty())

{

ve.push\_back(st.top());//反转放入vector

st.pop();

}

n-=k;

}while(n>=k);

while(n)//n>0且不足k个的直接放入vector

{

node2 p;

p.data=mp[head].data;

p.id=head;

p.next=mp[head].next;

ve.push\_back(p);

if(p.next==-1)break;

else head=p.next;

}

for(int i=0;i<ve.size();i++)//输出

{

printf("%05d %d ",ve[i].id,ve[i].data);

if(i==ve.size()-1)

{

printf("-1\n");

}

else

{

printf("%05d\n",ve[i+1].id);

}

}

return 0;

}

/\*\*

00100 4 4

00100 3 12345

12345 1 12121

12121 5 13131

13131 6 -1

00100 1 1

00100 1 -1

\*/

/\*\*

大佬的简短代码

#include <iostream>

#include <algorithm>

using namespace std;

int main(int argc, const char \* argv[])

{

int FirstAdd,Total,K,key;

cin>>FirstAdd>>Total>>K;

key=FirstAdd;

int temp,Data[100005],Next[100005];

int i,j;

for(i=0;i<Total;i++)

{

cin>>temp>>Data[temp]>>Next[temp];

}

int Order[100005],count=0;

while(key!=-1)

{

Order[count]=key;

key=Next[key];

count++;

}

int round=count/K;

for(i=j=0;j<round;i=i+K,j++)

{

reverse(Order+i, Order+i+K);

}

for(i=0;i<count-1;i++)

{

printf("%05d %d %05d\n",Order[i],Data[Order[i]],Order[i+1]);

}

printf("%05d %d -1\n",Order[i],Data[Order[i]]);

return 0;

}

1026 程序运行时间(15)

要获得一个 C 语言程序的运行时间，常用的方法是调用头文件 time.h，其中提供了 clock() 函数，可以捕捉从程序开始运行到 clock() 被调用时所耗费的时间。这个时间单位是 clock tick，即“时钟打点”。同时还有一个常数 CLK\_TCK，给出了机器时钟每秒所走的时钟打点数。于是为了获得一个函数 *f* 的运行时间，我们只要在调用 *f*之前先调用 clock()，获得一个时钟打点数 C1；在 *f* 执行完成后再调用 clock()，获得另一个时钟打点数 C2；两次获得的时钟打点数之差 (C2-C1) 就是 *f* 运行所消耗的时钟打点数，再除以常数 CLK\_TCK，就得到了以秒为单位的运行时间。

这里不妨简单假设常数 CLK\_TCK 为 100。现给定被测函数前后两次获得的时钟打点数，请你给出被测函数运行的时间。

### 输入格式：

输入在一行中顺序给出 2 个整数 C1 和 C2。注意两次获得的时钟打点数肯定不相同，即 C1 < C2，并且取值在 [0,10​7​​]。

### 输出格式：

在一行中输出被测函数运行的时间。运行时间必须按照 hh:mm:ss（即2位的 时:分:秒）格式输出；不足 1 秒的时间四舍五入到秒。

### 输入样例：

123 4577973

### 输出样例：

12:42:59

思路：

简单水题

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a,b,time;

cin>>a>>b;

time=(b-a)\*1.0/100+0.5;

int m=time/60;

int h=m/60;

m%=60;time%=60;

printf("%02d:%02d:%02d\n",h,m,time);

return 0;

}

1027 打印沙漏(20)

本题要求你写个程序把给定的符号打印成沙漏的形状。例如给定17个“\*”，要求按下列格式打印

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

所谓“沙漏形状”，是指每行输出奇数个符号；各行符号中心对齐；相邻两行符号数差2；符号数先从大到小顺序递减到1，再从小到大顺序递增；首尾符号数相等。

给定任意N个符号，不一定能正好组成一个沙漏。要求打印出的沙漏能用掉尽可能多的符号。

### 输入格式:

输入在一行给出1个正整数N（≤1000）和一个符号，中间以空格分隔。

### 输出格式:

首先打印出由给定符号组成的最大的沙漏形状，最后在一行中输出剩下没用掉的符号数。

### 输入样例:

19 \*

### 输出样例:

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

2

思路：

首先计算最多可以打印多大的沙漏，然后控制好每一行打印多少个空格和字符就好

代码：

#include <bits/stdc++.h>

using namespace std;

void out\_char(int n,char ch)

{

while(n--)printf("%c",ch);

}

int main()

{

int n,i,sum=0;

char ch;

scanf("%d %c",&n,&ch);

for(i=1;;i++)//计算最大行数

{

sum+=i\*2-1;

if(sum\*2-1>n)

{i--;sum=0;break;}

}

for(int j=0;j<i-1;j++)//打印上半个三角

{

out\_char(j,' ');

out\_char((i-j)\*2-1,ch);

sum+=(i-j)\*2-1;

cout<<endl;

}

for(int j=i-1;j>=0;j--)//下半个三角

{

out\_char(j,' ');

out\_char((i-j)\*2-1,ch);

sum+=(i-j)\*2-1;

cout<<endl;

}

cout<<n-sum<<endl;//输出剩余

return 0;

}

1028 人口普查(20)

某城镇进行人口普查，得到了全体居民的生日。现请你写个程序，找出镇上最年长和最年轻的人。

这里确保每个输入的日期都是合法的，但不一定是合理的——假设已知镇上没有超过 200 岁的老人，而今天是 2014 年 9 月 6 日，所以超过 200 岁的生日和未出生的生日都是不合理的，应该被过滤掉。

### 输入格式：

输入在第一行给出正整数 *N*，取值在(0,10​5​​]；随后 *N* 行，每行给出 1 个人的姓名（由不超过 5 个英文字母组成的字符串）、以及按 yyyy/mm/dd（即年/月/日）格式给出的生日。题目保证最年长和最年轻的人没有并列。

### 输出格式：

在一行中顺序输出有效生日的个数、最年长人和最年轻人的姓名，其间以空格分隔。

### 输入样例：

5

John 2001/05/12

Tom 1814/09/06

Ann 2121/01/30

James 1814/09/05

Steve 1967/11/20

### 输出样例：

3 Tom John

思路：

把日期转换成int类型直接判断范围和相减判断年龄大小就可以了

牛客过了，官网一个wa一个pe，wa在于年龄边界。pe是因为如果没有符合条件的则只输出0。

代码：

#include <bits/stdc++.h>

#define XDATE 20140906

#define MDATA 18140906

using namespace std;

int main()

{

string mins,maxs,ans;

int minn=XDATE,maxx=0,y,m,d,data,sum=0;

int n;cin>>n;

while(n--)

{

cin>>ans;

scanf("%d/%d/%d",&y,&m,&d);

data=y\*10000+m\*100+d;

if(data<=XDATE&&data>=MDATA)

{

sum++;

if(data>maxx)

{maxx=data;maxs=ans;}

if(data<minn)

{minn=data;mins=ans;}

}

}

if(sum)

cout<<sum<<' '<<mins<<' '<<maxs<<endl;

else

cout<<0<<endl;

return 0;

}

1029 旧键盘 (20)

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及实际被输入的文字，请你列出肯定坏掉的那些键。

### 输入格式：

输入在 2 行中分别给出应该输入的文字、以及实际被输入的文字。每段文字是不超过 80 个字符的串，由字母 A-Z（包括大、小写）、数字 0-9、以及下划线 \_（代表空格）组成。题目保证 2 个字符串均非空。

### 输出格式：

按照发现顺序，在一行中输出坏掉的键。其中英文字母只输出大写，每个坏键只输出一次。题目保证至少有 1 个坏键。

### 输入样例：

7\_This\_is\_a\_test

\_hs\_s\_a\_es

### 输出样例：

7TI

思路：

先储存键盘输出的字符，然后将应该打出的字符进行比较，如果没有输出过且没有被标记就变成大写输出来，同时标记一下这个字母的大小写。

牛客过了官网没过，原因想当然的认为了范围直接用else来做另一种情况少考虑的一种情况。要把每种情况都用if判断出来

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

char s[100],s2[100];

set<char>se;

map<char ,int >flag;

gets(s);gets(s2);

int l=strlen(s2);

for(int i=0;i<l;i++)

{

se.insert(s2[i]);

}

l=strlen(s);

for(int i=0;i<l;i++)

{

if(!se.count(s[i])&&!flag[s[i]])

{

flag[s[i]]=1;

if(s[i]>='a'&&s[i]<='z')

{

printf("%c",s[i]-32);

flag[s[i]-32]=1;

}

else if(s[i]>='A'&&s[i]<='Z')

{

flag[s[i]+32]=1;

printf("%c",s[i]);

}

else

{

printf("%c",s[i]);

}

}

}

cout<<endl;

return 0;

}

1030 完美数列(25)

给定一个正整数数列，和正整数 *p*，设这个数列中的最大值是 *M*，最小值是 *m*，如果 *M*≤*mp*，则称这个数列是完美数列。

现在给定参数 *p* 和一些正整数，请你从中选择尽可能多的数构成一个完美数列。

### 输入格式：

输入第一行给出两个正整数 *N* 和 *p*，其中 *N*（≤10​5​​）是输入的正整数的个数，*p*（≤10​9​​）是给定的参数。第二行给出 *N* 个正整数，每个数不超过 10​9​​。

### 输出格式：

在一行中输出最多可以选择多少个数可以用它们组成一个完美数列。

### 输入样例：

10 8

2 3 20 4 5 1 6 7 8 9

### 输出样例：

8

思路：

想了一会儿发现并没有什么好的思路，那就暴力找+二分优化，复杂度nlngn也可以接受，不过每次写二分的时候都会在边界问题上纠结好久，还是内功太差

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

long long n,p,a[100005],maxx=0;

scanf("%lld %lld",&n,&p);

for(int i=0;i<n;i++)

scanf("%lld",&a[i]);

sort(a,a+n);

for(int i=0;i<n-maxx;i++)

{

int ans=upper\_bound(a+i,a+n,a[i]\*p)-a;//因为查找到的是第一个大于查找值的地址，它的下一个就是<=a[i]\*p了

if(ans-i>maxx)//然后ans-i就是第i个到ans个之间的数量

maxx=ans-i;

}

cout<<maxx<<endl;

return 0;

}

1031 查验身份证(15)

一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；然后将计算的和对11取模得到值Z；最后按照以下关系对应Z值与校验码M的值：

Z：0 1 2 3 4 5 6 7 8 9 10

M：1 0 X 9 8 7 6 5 4 3 2

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

### 输入格式：

输入第一行给出正整数*N*（≤100）是输入的身份证号码的个数。随后*N*行，每行给出1个18位身份证号码。

### 输出格式：

按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出All passed。

### 输入样例1：

4

320124198808240056

12010X198901011234

110108196711301866

37070419881216001X

### 输出样例1：

12010X198901011234

110108196711301866

37070419881216001X

### 输入样例2：

2

320124198808240056

110108196711301862

### 输出样例2：

All passed

思路：

按照题意判断即可

代码：

#include <bits/stdc++.h>

using namespace std;

int quan[]={7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};

char jiao[]={'1','0','X','9','8','7','6','5','4','3','2'};

int judeg(char s[])

{

int sum=0;

for(int i=0;i<17;i++)

if(s[i]<'0'||s[i]>'9')

return 0;

else

{

sum+=(s[i]-'0')\*quan[i];

}

sum%=11;

if(jiao[sum]==s[17])

return 1;

return 0;

}

int main()

{

char id[20];

int n,flag=1;

cin>>n;getchar();

while(n--)

{

scanf("%s",id);

if(!judeg(id))

{

flag=0;

printf("%s\n",id);

}

}

if(flag)printf("All passed\n");

return 0;

}

1032 挖掘机技术哪家强(20)

思路：

存下每个学校id，然后遍历一遍找最大就好。

代码：

#include <bits/stdc++.h>

using namespace std;

int xue[100005];

int main()

{

set<int>se;

int n,xveid,maxx=0;

cin>>n;

while(n--)

{

int id,fen;

scanf("%d %d",&id,&fen);

se.insert(id);

xue[id]+=fen;

}

for(set<int>::iterator it=se.begin();it!=se.end();it++)

{

if(xue[\*it]>maxx)

{

xveid=\*it;

maxx=xue[\*it];

}

}

cout<<xveid<<' '<<maxx<<endl;

return 0;

}

1033 旧键盘打字(20)

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及坏掉的那些键，打出的结果文字会是怎样？

### 输入格式：

输入在 2 行中分别给出坏掉的那些键、以及应该输入的文字。其中对应英文字母的坏键以大写给出；每段文字是不超过 10​5​​ 个字符的串。可用的字符包括字母 [a-z, A-Z]、数字 0-9、以及下划线 \_（代表空格）、,、.、-、+（代表上档键）。题目保证第 2 行输入的文字串非空。

注意：如果上档键坏掉了，那么大写的英文字母无法被打出。

### 输出格式：

在一行中输出能够被打出的结果文字。如果没有一个字符能被打出，则输出空行。

### 输入样例：

7+IE.

7\_This\_is\_a\_test.

### 输出样例：

\_hs\_s\_a\_tst

思路：

存下坏的键，然后将要打出的字符一个个判断如果没坏就输出，否则不操作

牛客过官网wa一组，注意最后一句保证第二组非空，所以还有一种第行为空的情况

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int mp[300]={0},l;

char s[40],s2[100005];

gets(s);

gets(s2);

for(int i=0;i<strlen(s);i++)

{

if(s[i]>='A'&&s[i]<='Z')

{

mp[s[i]]=1;

mp[s[i]+32]=1;

}

else

mp[s[i]]=1;

}

l=strlen(s2);

for(int i=0;i<l;i++)

{

if(s2[i]>='A'&&s2[i]<='Z')

{

if(!mp['+']&&!mp[s2[i]])

printf("%c",s2[i]);

}

else

{

if(!mp[s2[i]])

printf("%c",s2[i]);

}

}

cout<<endl;

return 0;

}

/\*\*

7+IE.

7\_This\_is\_a\_test.

\*/

1034 有理数四则运算(20)

### 输入格式：

输入在一行中按照 a1/b1 a2/b2 的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的整数，负号只可能出现在分子前，分母不为 0。

### 输出格式：

分别在 4 行中按照 有理数1 运算符 有理数2 = 结果 的格式顺序输出 2 个有理数的和、差、积、商。注意输出的每个有理数必须是该有理数的最简形式 k a/b，其中 k 是整数部分，a/b 是最简分数部分；若为负数，则须加括号；若除法分母为 0，则输出 Inf。题目保证正确的输出中没有超过整型范围的整数。

### 输入样例 1：

2/3 -4/2

### 输出样例 1：

2/3 + (-2) = (-1 1/3)

2/3 - (-2) = 2 2/3

2/3 \* (-2) = (-1 1/3)

2/3 / (-2) = (-1/3)

### 输入样例 2：

5/3 0/6

### 输出样例 2：

1 2/3 + 0 = 1 2/3

1 2/3 - 0 = 1 2/3

1 2/3 \* 0 = 0

1 2/3 / 0 = Inf

思路：

用四个变量分别储存分子分明，模拟分数的计算方法，然后写好化简的函数将数字转换成最简输出

牛客过官网wa，原因忘记考虑两个都为负数的情况

代码：

#include <bits/stdc++.h>

#define ull long long

using namespace std;

ull gcd(ull a,ull b)

{

return !(a%b)?b:gcd(b,a%b);

}

void huajian(ull fz,ull fm)

{

ull gy,flag=0;

if(fz<0||fm<0)

{

if(!(fz<0&&fm<0))

{

printf("(-");

flag=1;

}

if(fz<0)fz=-fz;

if(fm<0)fm=-fm;

}

gy=gcd(fz,fm);

fz/=gy;fm/=gy;

if(!fz)

{

printf("0");

}

else if(fm==1)

{

printf("%lld",fz);

}

else if(fz>fm)

{

printf("%lld %lld/%lld",fz/fm,fz%fm,fm);

}

else

printf("%lld/%lld",fz%fm,fm);

if(flag)

printf(")");

}

int main()

{

ull a,b,c,d;

scanf("%lld/%lld %lld/%lld",&a,&b,&c,&d);

huajian(a,b);

printf(" + ");

huajian(c,d);

printf(" = ");

huajian(a\*d+b\*c,b\*d);

printf("\n");

huajian(a,b);

printf(" - ");

huajian(c,d);

printf(" = ");

huajian(a\*d-b\*c,b\*d);

printf("\n");

huajian(a,b);

printf(" \* ");

huajian(c,d);

printf(" = ");

huajian(a\*c,b\*d);

printf("\n");

huajian(a,b);

printf(" / ");

huajian(c,d);

printf(" = ");

if(!c)

printf("Inf");

else

huajian(a\*d,b\*c);

printf("\n");

return 0;

}

1035 插入与归并(25)

**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？

### 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，且行首尾不得有多余空格。

### 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

思路：

没有什么好的思路，用插入和归并排一遍对比中间过程，如果遇到相同的就输出下一次排序的序列，当然想要A掉这一题还要会这两种排序，不会的话。。。跟我一样现学去吧。╮(╯\_╰)╭

牛客过了官网没过，5号到7号卡了三天，代码从95行写到39行，能用别的方法还是不要模拟的好，模拟细节太多了╮(╯\_╰)╭，现在对归并理解的又深了

代码：

#include <bits/stdc++.h>

using namespace std;

int yuanshi[105],mubiao[105],n,ans[105],flag=0;

int juedg(int a[],int b[],int l)

{

for(int i=0;i<l;i++)

{

if(a[i]!=b[i])

return 0;

}

return 1;

}

void mergepass(int a[],int b[],int k,int n)

{

if(flag==2)return ;

int i;

for(i=0;i<=n-k\*2;i+=k\*2)

{

merge(a+i,a+i+k,a+i+k,a+i+2\*k,b+i);

}

if(n<k\*2)

merge(a,a+k,a+k,a+n,b);

if(flag)

{

for(int i=0;i<n;i++)

{

printf("%d%c",b[i],i==n-1?'\n':' ');

}

flag=2;

return ;

}

if(juedg(mubiao,b,n))

{

printf("Merge Sort\n");

flag=1;

}

}

void input()

{

for(int i=0;i<n;i++)

scanf("%d",&yuanshi[i]);

for(int i=0;i<n;i++)

scanf("%d",&mubiao[i]);

}

int main()

{

cin>>n;

input();

copy(yuanshi,yuanshi+n,ans);

for(int i=1;i<n;i++)//插入排序

{

if(ans[i]<ans[i-1])

{

int p=upper\_bound(ans,ans+i,ans[i])-ans,q=ans[i];

copy(ans+p,ans+i,ans+p+1);

ans[p]=q;

if(flag)

{

for(int i=0;i<n;i++)

{

printf("%d%c",ans[i],i==n-1?'\n':' ');

}

return 0;

}

if(juedg(mubiao,ans,n))

{

printf("Insertion Sort\n");

flag=1;

}

}

}

int k=1;//归并排序

while(k<n)

{

mergepass(yuanshi,ans,k,n);

k<<=1;

mergepass(ans,yuanshi,k,n);

k<<=1;

if(flag==2)

break;

}

return 0;

}

1036 跟奥巴马一起编程(15)

美国总统奥巴马不仅呼吁所有人都学习编程，甚至以身作则编写代码，成为美国历史上首位编写计算机代码的总统。2014 年底，为庆祝“计算机科学教育周”正式启动，奥巴马编写了很简单的计算机代码：在屏幕上画一个正方形。现在你也跟他一起画吧！

### 输入格式：

输入在一行中给出正方形边长 *N*（3≤*N*≤20）和组成正方形边的某种字符 C，间隔一个空格。

### 输出格式：

输出由给定字符 C 画出的正方形。但是注意到行间距比列间距大，所以为了让结果看上去更像正方形，我们输出的行数实际上是列数的 50%（四舍五入取整）。

### 输入样例：

10 a

### 输出样例：

aaaaaaaaaa

a a

a a

a a

aaaaaaaaaa

思路：

水题一道，四舍五入的时候要细心

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

char ch;

cin>>n>>ch;

for(int i=0;i<n;i++)

cout<<ch;

cout<<endl;

for(int j=0;j<(int)(n/2.0+0.5)-2;j++)

{

for(int i=0;i<n;i++)

printf("%c",i==0||i==n-1?ch:' ');

cout<<endl;

}

for(int i=0;i<n;i++)

cout<<ch;

cout<<endl;

return 0;

}

1037 在霍格沃茨找零钱（20）

### 输入格式：

输入在 1 行中分别给出 *P* 和 *A*，格式为 Galleon.Sickle.Knut，其间用 1 个空格分隔。这里 Galleon 是 [0, 10​7​​] 区间内的整数，Sickle 是 [0, 17) 区间内的整数，Knut 是 [0, 29) 区间内的整数。

### 输出格式：

在一行中用与输入同样的格式输出哈利应该被找的零钱。如果他没带够钱，那么输出的应该是负数。

### 输入样例 1：

10.16.27 14.1.28

### 输出样例 1：

3.2.1

### 输入样例 2：

14.1.28 10.16.27

### 输出样例 2：

-3.2.1

思路：

先全部转换成最小的单位，然后相减在转换回来，另类的进制转换

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int G,S,K,GSK\_sum=0,g,s,k,gsk\_sum=0;

scanf("%d.%d.%d %d.%d.%d",&g,&s,&k,&G,&S,&K);

GSK\_sum=(G\*17+S)\*29+K;

gsk\_sum=(g\*17+s)\*29+k;

GSK\_sum-=gsk\_sum;

if(GSK\_sum<0){GSK\_sum=-GSK\_sum;printf("-");}

K=GSK\_sum%29;

S=(GSK\_sum/29)%17;

G=(GSK\_sum/29)/17;

printf("%d.%d.%d\n",G,S,K);

return 0;

}

1038 统计同成绩学生(20)

### 输入格式：

输入在第 1 行给出不超过 10​5​​ 的正整数 *N*，即学生总人数。随后一行给出 *N* 名学生的百分制整数成绩，中间以空格分隔。最后一行给出要查询的分数个数 *K*（不超过 *N* 的正整数），随后是 *K* 个分数，中间以空格分隔。

### 输出格式：

在一行中按查询顺序给出得分等于指定分数的学生人数，中间以空格分隔，但行末不得有多余空格。

### 输入样例：

10

60 75 90 55 75 99 82 90 75 50

3 75 90 88

### 输出样例：

3 2 0

思路：

下标法标记，水题

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int mp[105]={0},ans;

int n;cin>>n;

for(int i=0;i<n;i++)

{

scanf("%d",&ans);

mp[ans]++;

}

cin>>n;

while(n--)

{

scanf("%d",&ans);

printf("%d%c",mp[ans],n?' ':'\n');

}

return 0;

}

1039 到底买不买（20）

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。

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。



图 1

### 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。

### 输入样例 1：

ppRYYGrrYBR2258

YrR8RrY

### 输出样例 1：

Yes 8

### 输入样例 2：

ppRYYGrrYB225

YrR8RrY

### 输出样例 2：

No 2

思路：

存下每个需要的珠子的数量，然后遍历另一个串，查看是否都有，如果所需要珠子都有输出l-l2，否则遍历标记数组统计有多少没有，（刚开始我还想了有没有需要的珠子有多出的情况这时候多余的珠子就不再是l-l2，后来测试发现没有这样的数据，这个问题考不考虑都能A）

代码：

#include <bits/stdc++.h>

using namespace std;

int judeg(int a[])

{

int sum=0;

for(int i=0;i<300;i++)

if(a[i])sum+=a[i];

return sum;

}

int main()

{

int mp[300]={0},sum=0,l,l2;

char s[1005],s2[1005];

scanf("%s %s",s,s2);

l=strlen(s);sum=l2=strlen(s2);

for(int i=0;i<l2;i++)

{

mp[s2[i]]++;

}

for(int i=0;i<l;i++)

{

if(mp[s[i]])

{

sum--;

mp[s[i]]--;

}

}

if(!judeg(mp))

{

printf("Yes %d\n",l-l2+sum);

}

else

{

printf("No %d\n",judeg(mp));

}

return 0;

}

/\*\*

ppRYYGrrYBR2258

YrR8RrY

\*/

1040 有几个PAT（25）

字符串 APPAPT 中包含了两个单词 PAT，其中第一个 PAT 是第 2 位(P)，第 4 位(A)，第 6 位(T)；第二个 PAT 是第 3 位(P)，第 4 位(A)，第 6 位(T)。

现给定字符串，问一共可以形成多少个 PAT？

### 输入格式：

输入只有一行，包含一个字符串，长度不超过10​5​​，只包含 P、A、T 三种字母。

### 输出格式：

在一行中输出给定字符串中包含多少个 PAT。由于结果可能比较大，只输出对 1000000007 取余数的结果。

### 输入样例：

APPAPT

### 输出样例：

2

思路：

这个思路有点儿动态规划的感觉，每次遇到P将数量++，碰到A的时候将pa数量加上P的数量（就是这个A可以组合出来的Pa的数量），每碰到一个T的时候将PA的数量累加的SUM上，这是这个T可以组合出来的所有PAT，然后每一步判断是否要取余，最后答案就出来了

代码：

#include <bits/stdc++.h>

#define MOD 1000000007

using namespace std;

int main()

{

char ch;

int p=0,a=0,sum=0;

while(~scanf("%c",&ch))

{

if(ch=='P')p++;

else if(ch=='A')a+=p;

else if(ch=='T')

{

sum+=a;

if(sum>MOD)

sum%=MOD;

}

}

cout<<sum<<endl;

return 0;

}

1041 考试座位号(15)

每个 PAT 考生在参加考试时都会被分配两个座位号，一个是试机座位，一个是考试座位。正常情况下，考生在入场时先得到试机座位号码，入座进入试机状态后，系统会显示该考生的考试座位号码，考试时考生需要换到考试座位就座。但有些考生迟到了，试机已经结束，他们只能拿着领到的试机座位号码求助于你，从后台查出他们的考试座位号码。

### 输入格式：

输入第一行给出一个正整数 *N*（≤1000），随后 *N* 行，每行给出一个考生的信息：准考证号 试机座位号 考试座位号。其中准考证号由 16 位数字组成，座位从 1 到 *N* 编号。输入保证每个人的准考证号都不同，并且任何时候都不会把两个人分配到同一个座位上。

考生信息之后，给出一个正整数 *M*（≤*N*），随后一行中给出 *M* 个待查询的试机座位号码，以空格分隔。

### 输出格式：

对应每个需要查询的试机座位号码，在一行中输出对应考生的准考证号和考试座位号码，中间用 1 个空格分隔。

### 输入样例：

4

3310120150912233 2 4

3310120150912119 4 1

3310120150912126 1 3

3310120150912002 3 2

2

3 4

### 输出样例：

3310120150912002 2

3310120150912119 1

思路：

水题

代码：

#include <bits/stdc++.h>

using namespace std;

struct node

{

string id;

int zw;

}a[1005];

int main()

{

int n,sj,zw;

string s;

cin>>n;

while(n--)

{

cin>>s>>sj>>zw;

a[sj].id=s;

a[sj].zw=zw;

}

cin>>n;

while(n--)

{

cin>>sj;

cout<<a[sj].id<<' '<<a[sj].zw<<endl;

}

return 0;

}

1042 字符统计(20)

请编写程序，找出一段给定文字中出现最频繁的那个英文字母。

### 输入格式：

输入在一行中给出一个长度不超过 1000 的字符串。字符串由 ASCII 码表中任意可见字符及空格组成，至少包含 1 个英文字母，以回车结束（回车不算在内）。

### 输出格式：

在一行中输出出现频率最高的那个英文字母及其出现次数，其间以空格分隔。如果有并列，则输出按字母序最小的那个字母。统计时不区分大小写，输出小写字母。

### 输入样例：

This is a simple TEST. There ARE numbers and other symbols 1&2&3...........

### 输出样例：

e 7

思路：

水题

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int mp[30]={0},maxx=0;

char ch;

while(scanf("%c",&ch),ch!='\n')

{

if(ch>='A'&&ch<='Z')

ch+=32;

if(ch>='a'&&ch<='z')

mp[ch-'a']++;

}

for(int i=0;i<26;i++)

{

if(mp[i]>mp[maxx])

maxx=i;

}

printf("%c %d\n",maxx+'a',mp[maxx]);

return 0;

}

1043 输出PATest(20)

给定一个长度不超过 10​4​​ 的、仅由英文字母构成的字符串。请将字符重新调整顺序，按 PATestPATest.... 这样的顺序输出，并忽略其它字符。当然，六种字符的个数不一定是一样多的，若某种字符已经输出完，则余下的字符仍按 PATest 的顺序打印，直到所有字符都被输出。

### 输入格式：

输入在一行中给出一个长度不超过 10​4​​ 的、仅由英文字母构成的非空字符串。

### 输出格式：

在一行中按题目要求输出排序后的字符串。题目保证输出非空。

### 输入样例：

redlesPayBestPATTopTeePHPereatitAPPT

### 输出样例：

PATestPATestPTetPTePePee

思路：

水，照着题目意思来就好

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int mp[30]={0},maxx=0;

char ch;

while(scanf("%c",&ch),ch!='\n')

{

if(ch>='A'&&ch<='Z')

ch+=32;

if(ch>='a'&&ch<='z')

mp[ch-'a']++;

}

for(int i=0;i<26;i++)

{

if(mp[i]>mp[maxx])

maxx=i;

}

printf("%c %d\n",maxx+'a',mp[maxx]);

return 0;

}

1044 火星数字(20)

火星人是以 13 进制计数的：

* 地球人的 0 被火星人称为 tret。
* 地球人数字 1 到 12 的火星文分别为：jan, feb, mar, apr, may, jun, jly, aug, sep, oct, nov, dec。
* 火星人将进位以后的 12 个高位数字分别称为：tam, hel, maa, huh, tou, kes, hei, elo, syy, lok, mer, jou。

例如地球人的数字 29 翻译成火星文就是 hel mar；而火星文 elo nov 对应地球数字 115。为了方便交流，请你编写程序实现地球和火星数字之间的互译。

输入格式：

输入第一行给出一个正整数 *N*（<100），随后 *N* 行，每行给出一个 [0, 169) 区间内的数字 —— 或者是地球文，或者是火星文。

输出格式：

对应输入的每一行，在一行中输出翻译后的另一种语言的数字。

输入样例：

4

29

5

elo nov

tam

输出样例：

hel mar

may

115

13

思路：

这个题目的意思实现起来并不难，不过被坑了好久，我是被坑在高低位的火星文是不一样的，如果只有一位火星文的时候我默认当作低位来算了，事实上只有一位要判断是高位还是低位在计算。

代码：

#include <bits/stdc++.h>

using namespace std;

string \_12q[]= {"tret", "jan","feb","mar","apr","may","jun","jly","aug","sep","oct","nov","dec"};

string \_12h[]= {"tret", "tam","hel","maa","huh","tou","kes","hei","elo","syy","lok","mer","jou"};

int stio(string ans)

{

for(int i=0; i<13; i++)

{

if(\_12q[i]==ans)

return i;

}

for(int i=0; i<13; i++)

{

if(\_12h[i]==ans)

return i\*13;

}

return 0;

}

int main()

{

int n,num,t=0;

cin>>n;getchar();

cin.clear();

while(n--)

{

string ans;

getline(cin,ans);

stringstream ss(ans);

if(ans[0]<='9'&&ans[0]>='0')

{

ss>>num;

if(num<13)

cout<<\_12q[num]<<endl;

else

{

t=num%13;

num/=13;

cout<<\_12h[num];

if(t)

cout<<' '<<\_12q[t];

cout<<endl;

}

}

else

{

ss>>ans;

t=0;

num=stio(ans);

while(ss>>ans)

{t = stio(ans);}

cout<<num+t<<endl;

ss.clear();

}

}

return 0;

}

1045 快速排序(25)

著名的快速排序算法里有一个经典的划分过程：我们通常采用某种方法取一个元素作为主元，通过交换，把比主元小的元素放到它的左边，比主元大的元素放到它的右边。 给定划分后的 *N* 个互不相同的正整数的排列，请问有多少个元素可能是划分前选取的主元？

例如给定 $N = 5$, 排列是1、3、2、4、5。则：

* 1 的左边没有元素，右边的元素都比它大，所以它可能是主元；
* 尽管 3 的左边元素都比它小，但其右边的 2 比它小，所以它不能是主元；
* 尽管 2 的右边元素都比它大，但其左边的 3 比它大，所以它不能是主元；
* 类似原因，4 和 5 都可能是主元。

因此，有 3 个元素可能是主元。

输入格式：

输入在第 1 行中给出一个正整数 *N*（≤10​5​​）； 第 2 行是空格分隔的 *N* 个不同的正整数，每个数不超过 10​9​​。

输出格式：

在第 1 行中输出有可能是主元的元素个数；在第 2 行中按递增顺序输出这些元素，其间以 1 个空格分隔，行首尾不得有多余空格。

输入样例：

5

1 3 2 4 5

输出样例：

3

1 4 5

思路：

我的思路比较low 一个原数组一个排好序的数组，存下左边的最大值，右边排好序保持第一个就是原数组中右边的最小值，这样比下去把满足条件的存下来。AC后看其他人代码发现，还是没有找到排序的精髓，主元的位置就是拍好的位置，所以说遍历两个数字碰到相同的且这个数字是左边最大的就是主元（这样说不知道清不清楚意思），按照这个思路代码就简单好多了，果然还是思路最重要。因为数据比较大直接硬算会超时，我那个方法算是一种取巧的硬算。

代码：

#include <bits/stdc++.h>

#define INF 1000000005

using namespace std;

int main()

{

int n,a[100005],j=0,b[100005],maxx=-1,minn=0;

map<int,int>mp;

vector<int>ve;

cin>>n;

for(int i=0;i<n;i++)

{

scanf("%d",&a[i]);

b[i]=a[i];

mp[a[i]]++;

}

b[n]=INF;mp[INF]=1;

sort(b,b+n+1);

for(int i=0;i<n;i++)

{

mp[a[i]]--;

while(!mp[b[maxx]]) maxx++;

//printf("%d %d %d---max==%d\n",minn,a[i],b[maxx],maxx);

if(a[i]>minn&&a[i]<b[maxx])

{

ve.push\_back(a[i]);

}

minn=max(a[i],minn);

}

sort(ve.begin(),ve.end());

cout<<ve.size()<<endl;

if(ve.size()) cout<<ve[0];

for(int i=1;i<ve.size();i++)

{

printf(" %d",ve[i]);

}

cout<<endl;

return 0;

}

1046 划拳(15)

思路：

水题，直接比大小

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a,b,c,d,n,j=0,y=0;

cin>>n;

while(n--)

{

cin>>a>>b>>c>>d;

int ans=a+c;

if(ans==b&&ans!=d) y++;

else if(ans!=b&&ans==d) j++;

}

cout<<j<<' '<<y<<endl;

return 0;

}

1047. 编程团体赛(20)

编程团体赛的规则为：每个参赛队由若干队员组成；所有队员独立比赛；参赛队的成绩为所有队员的成绩和；成绩最高的队获胜。

现给定所有队员的比赛成绩，请你编写程序找出冠军队。

### 输入格式：

输入第一行给出一个正整数 *N*（≤10​4​​），即所有参赛队员总数。随后 *N* 行，每行给出一位队员的成绩，格式为：队伍编号-队员编号 成绩，其中队伍编号为 1 到 1000 的正整数，队员编号为 1 到 10 的正整数，成绩为 0 到 100 的整数。

### 输出格式：

在一行中输出冠军队的编号和总成绩，其间以一个空格分隔。注意：题目保证冠军队是唯一的。

### 输入样例：

6

3-10 99

11-5 87

102-1 0

102-3 100

11-9 89

3-2 61

### 输出样例：

11 176

思路：

开一个下标为队伍的数组，直接累加储存最大值即可。

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,d,f,dui[1005]={0},maxd=0,maxx=0;

cin>>n;

while(n--)

{

scanf("%d-%\*d %d",&d,&f);

dui[d]+=f;

if(dui[d]>maxx)

{

maxd=d;maxx=dui[d];

}

}

cout<<maxd<<' '<<maxx<<endl;

return 0;

}

1048 数字加密(20)

本题要求实现一种数字加密方法。首先固定一个加密用正整数 A，对任一正整数 B，将其每 1 位数字与 A 的对应位置上的数字进行以下运算：对奇数位，对应位的数字相加后对 13 取余——这里用 J 代表 10、Q 代表 11、K 代表 12；对偶数位，用 B 的数字减去 A 的数字，若结果为负数，则再加 10。这里令个位为第 1 位。

### 输入格式：

输入在一行中依次给出 A 和 B，均为不超过 100 位的正整数，其间以空格分隔。

### 输出格式：

在一行中输出加密后的结果。

### 输入样例：

1234567 368782971

### 输出样例：

3695Q8118

思路：

字符串处理，不过有两个坑点，两个字符串长度不一致的时候要补0（这个测试数据里面有两个），二、注意前导零和处理后全为零的情况（这个测试数据中没有）

代码：

#include <bits/stdc++.h>

using namespace std;

char ji(char a,char b)

{

int ans=a+b-'0'\*2;

if(ans%13<10)return ans%13+'0';

else if(ans==10) return 'J';

else if(ans==11) return 'Q';

else if(ans==12) return 'K';

}

char ou(char a,char b)

{

if(b-a>=0) return b-a+'0';

else return b-a+10+'0';

}

int main()

{

char a[105],b[105],l,l2,ll;

scanf("%s %s",a,b);

l=strlen(a);l2=strlen(b);

reverse(a,a+l);reverse(b,b+l2);

if(l<l2)

{

ll=l2;

for(;l<l2;l++)

a[l]='0';

a[l]=0;

}

else

{

ll=l;

for(;l2<l;l2++)

b[l2]='0';

b[l2]=0;

}

for(int i=0;i<ll;i++)

{

if(i%2)

b[i]=ou(a[i],b[i]);

else

b[i]=ji(a[i],b[i]);

}

reverse(b,b+ll);

l=0;

while(b[l]=='0')l++;

if(strlen(b)==l)

printf("0\n");

else

printf("%s\n",b+l);

return 0;

}

1049 数列的片段和(20)

给定一个正数数列，我们可以从中截取任意的连续的几个数，称为片段。例如，给定数列 { 0.1, 0.2, 0.3, 0.4 }，我们有 (0.1) (0.1, 0.2) (0.1, 0.2, 0.3) (0.1, 0.2, 0.3, 0.4) (0.2) (0.2, 0.3) (0.2, 0.3, 0.4) (0.3) (0.3, 0.4) (0.4) 这 10 个片段。

给定正整数数列，求出全部片段包含的所有的数之和。如本例中 10 个片段总和是 0.1 + 0.3 + 0.6 + 1.0 + 0.2 + 0.5 + 0.9 + 0.3 + 0.7 + 0.4 = 5.0。

### 输入格式：

输入第一行给出一个不超过 10​5​​ 的正整数 *N*，表示数列中数的个数，第二行给出 *N* 个不超过 1.0 的正数，是数列中的数，其间以空格分隔。

### 输出格式：

在一行中输出该序列所有片段包含的数之和，精确到小数点后 2 位。

### 输入样例：

4

0.1 0.2 0.3 0.4

### 输出样例：

5.00

思路：

数学题，如果直接遍历肯定GG，把它写下来我们不难找到规律，第i个数出现的次数是（n-i）\*（i+1）次，所以输入的时候直接计算就好。（注意：计算（n-i）\*（i+1）的时候如果用整型会溢出）

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,num=0;

double sum=0,ans;

cin>>n;

for(int i=0;i<n;i++)

{

scanf("%lf",&ans);

sum+=ans\*(n-i)\*(i+1);

}

printf("%.2lf\n",sum);

return 0;

}

1050 螺旋矩阵(25)

本题要求将给定的 *N* 个正整数按非递增的顺序，填入“螺旋矩阵”。所谓“螺旋矩阵”，是指从左上角第 1 个格子开始，按顺时针螺旋方向填充。要求矩阵的规模为 *m* 行 *n* 列，满足条件：*m*×*n* 等于 *N*；*m*≥*n*；且 *m*−*n* 取所有可能值中的最小值。

### 输入格式：

输入在第 1 行中给出一个正整数 *N*，第 2 行给出 *N* 个待填充的正整数。所有数字不超过 10​4​​，相邻数字以空格分隔。

### 输出格式：

输出螺旋矩阵。每行 *n* 个数字，共 *m* 行。相邻数字以 1 个空格分隔，行末不得有多余空格。

### 输入样例：

12

37 76 20 98 76 42 53 95 60 81 58 93

### 输出样例：

98 95 93

42 37 81

53 20 76

58 60 76

思路：

这个题目就是以前写过的蛇形填数，不过问题在于矩阵的长宽不知道。所以没有办法提前定义矩阵的大小，只能算出n和m后动态申请或者使用vector，如果N是一个很大的素数，那么矩阵的高就会接近10^4，这样如果提前定义了这么大的二维数组肯定要内存超限，矩阵问题解决后，剩下的排序和填数就容易了。

代码：

#include <bits/stdc++.h>

using namespace std;

int a[10005],n,m;

int cmp(int a,int b)

{

return a>b;

}

vector<vector<int> >mn;//动态二维数组

vector<int>mm;

int main()

{

int N;cin>>N;

for(int i=0;i<N;i++)

{

scanf("%d",&a[i]);

}

sort(a,a+N,cmp);

n=sqrt(N);

while(n>1&&N%n!=0)n--;

m=N/n;

for(int j=0;j<=n+1;j++)//先申请宽

mm.push\_back(-1);

for(int i=0;i<=m+1;i++)//再申请高

{

mn.push\_back(mm);

}

for(int k=0,i=0,j=0;k<N;)

{

for(i++,j++;j<=n&&mn[i][j]==-1&&k<N;j++,k++) mn[i][j]=a[k];

for(j--,i++;i<=m&&mn[i][j]==-1&&k<N;i++,k++) mn[i][j]=a[k];

for(i--,j--;j>0&&mn[i][j]==-1&&k<N;j--,k++) mn[i][j]=a[k];

for(j++,i--;i>0&&mn[i][j]==-1&&k<N;i--,k++) mn[i][j]=a[k];

}

for(int i=1;i<=m;i++)

for(int j=1;j<=n;j++)

printf("%d%c",mn[i][j],j==n?'\n':' ');

return 0;

}

1051 复数乘法 (15)

思路：

数学题，不过要注意保留两位小数的问题，%.2lf是四舍五入两位

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

double r1,r2,p1,p2;

cin>>r1>>p1>>r2>>p2;

double A=(r1\*r2)\*cos(p1+p2),B=(r1\*r2)\*sin(p1+p2);

if(abs(A)<0.01)A=0;

if(abs(B)<0.01)B=0;

printf("%.2lf%+.2lfi\n",A,B);

return 0;

}

1052 卖个萌 (20)

萌萌哒表情符号通常由“手”、“眼”、“口”三个主要部分组成。简单起见，我们假设一个表情符号是按下列格式输出的：

[左手]([左眼][口][右眼])[右手]

现给出可选用的符号集合，请你按用户的要求输出表情。

### 输入格式：

输入首先在前三行顺序对应给出手、眼、口的可选符号集。每个符号括在一对方括号 []内。题目保证每个集合都至少有一个符号，并不超过 10 个符号；每个符号包含 1 到 4 个非空字符。

之后一行给出一个正整数 K，为用户请求的个数。随后 K 行，每行给出一个用户的符号选择，顺序为左手、左眼、口、右眼、右手——这里只给出符号在相应集合中的序号（从 1 开始），数字间以空格分隔。

### 输出格式：

对每个用户请求，在一行中输出生成的表情。若用户选择的序号不存在，则输出 Are you kidding me? @\/@。

### 输入样例：

[╮][╭][o][~\][/~] [<][>]

[╯][╰][^][-][=][>][<][@][⊙]

[Д][▽][\_][ε][^] ...

4

1 1 2 2 2

6 8 1 5 5

3 3 4 3 3

2 10 3 9 3

### 输出样例：

╮(╯▽╰)╭

<(@Д=)/~

o(^ε^)o

Are you kidding me? @\/@

思路：

这道题目比较麻烦，因为每行会有空格什么的，所以要直接获取一行然后再处理将表情存起来，中括号里面的表情有中文字符的可能要涉及到字符编码问题，我第一次用字符数组转存的时候全变成了空字符串，后来该改成string，用substr()方法截取表情输出正常。不过有的电脑上个别符号显示不出来，提交开始可以AC的。

坑点：错误的那句有一个反斜杠需要转义

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

vector<string>ve[3];

for(int k=0; k<3; k++)

{

string s;

getline(cin,s);

for(int i=0; i<s.size(); i++)

{

if(s[i]=='[')

{

for(int j=i+1;j<s.size(); j++)

{

if(s[j]==']')

{

ve[k].push\_back(s.substr(i+1,j-i-1));

break;

}

}

}

}

}

int n,m[5];

cin>>n;

int hand=ve[0].size(),eye=ve[1].size(),mouth=ve[2].size();

while(n--)

{

for(int i=0; i<5; i++)

{

cin>>m[i];

m[i]--;

}

if(m[0]<hand&&m[1]<eye&&m[2]<mouth&&m[3]<eye&&m[4]<hand&&m[0]>=0&&m[1]>=0&&m[2]>=0&&m[3]>=0&&m[4]>=0)

{

cout<<ve[0][m[0]]<<'('<<ve[1][m[1]]<<ve[2][m[2]]<<ve[1][m[3]]<<')'<<ve[0][m[4]]<<endl;

}

else

{

cout<<"Are you kidding me? @\\/@"<<endl;

}

}

return 0;

}

1053 住房空置率 (20)

在不打扰居民的前提下，统计住房空置率的一种方法是根据每户用电量的连续变化规律进行判断。判断方法如下：

* 在观察期内，若存在超过一半的日子用电量低于某给定的阈值 *e*，则该住房为“可能空置”；
* 若观察期超过某给定阈值 *D* 天，且满足上一个条件，则该住房为“空置”。

现给定某居民区的住户用电量数据，请你统计“可能空置”的比率和“空置”比率，即以上两种状态的住房占居民区住房总套数的百分比。

### 输入格式：

输入第一行给出正整数 *N*（≤1000），为居民区住房总套数；正实数 *e*，即低电量阈值；正整数 *D*，即观察期阈值。随后 *N* 行，每行按以下格式给出一套住房的用电量数据：

*K* *E*​1​​ *E*​2​​ ... *E*​*K*​​

其中 *K* 为观察的天数，*E*​*i*​​ 为第 *i* 天的用电量。

### 输出格式：

在一行中输出“可能空置”的比率和“空置”比率的百分比值，其间以一个空格分隔，保留小数点后 1 位。

### 输入样例：

5 0.5 10

6 0.3 0.4 0.5 0.2 0.8 0.6

10 0.0 0.1 0.2 0.3 0.0 0.8 0.6 0.7 0.0 0.5

5 0.4 0.3 0.5 0.1 0.7

11 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

11 2 2 2 1 1 0.1 1 0.1 0.1 0.1 0.1

### 输出样例：

40.0% 20.0%

（样例解释：第2、3户为“可能空置”，第4户为“空置”，其他户不是空置。

思路：

看题啊看题，思路很简单题目信息看错了结果wa了两组，找半天最后发现是题目看错了

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,m,d,k=0,t=0;

double e,ans;

cin>>n>>e>>d;

for(int i=0;i<n;i++)

{

cin>>m;

int sum=0;

for(int j=0;j<m;j++)

{

cin>>ans;

if(ans<e) sum++;

}

if(sum>m/2) m>d?t++:k++;

}

printf("%.1lf%% %.1lf%%\n",k\*100.0/n,t\*100.0/n);

return 0;

}

1054 求平均值 (20)

思路：

又是读题没看k==1的情况 ，一定要养成仔细读题的习惯。

代码：

#include <bits/stdc++.h>

using namespace std;

int judeg(char s[])

{

int l=strlen(s),flag=0,k=l;

int i=s[0]=='-'?1:0;

for(;i<l;i++)

{

if((s[i]>='0'&&s[i]<='9')||s[i]=='.')

{if(s[i]=='.'){flag++;k=i;}}

else return 0;

}

if(flag>1||l-k>3)return 0;

return 1;

}

double AVG(char s[])

{

if(!judeg(s)) return 9999;

double n;

sscanf(s,"%lf",&n);

return n;

}

int main()

{

int n,m=0;

double avg,sum=0;

char s[105];

cin>>n;

while(n--)

{

scanf("%s",s);

avg=AVG(s);

if(avg<-1000||avg>1000)

printf("ERROR: %s is not a legal number\n",s);

else

{

m++; sum+=avg;

}

}

if(m==1) printf("The average of 1 number is ");

else printf("The average of %d numbers is ",m);

if(m) printf("%.2lf\n",sum/m);

else printf("Undefined\n");

return 0;

}

1055 集体照 (25)

拍集体照时队形很重要，这里对给定的 *N* 个人 *K* 排的队形设计排队规则如下：

* 每排人数为 *N*/*K*（向下取整），多出来的人全部站在最后一排；
* 后排所有人的个子都不比前排任何人矮；
* 每排中最高者站中间（中间位置为 *m*/2+1，其中 *m* 为该排人数，除法向下取整）；
* 每排其他人以中间人为轴，按身高非增序，先右后左交替入队站在中间人的两侧（例如5人身高为190、188、186、175、170，则队形为175、188、190、186、170。这里假设你面对拍照者，所以你的左边是中间人的右边）；
* 若多人身高相同，则按名字的字典序升序排列。这里保证无重名。

现给定一组拍照人，请编写程序输出他们的队形。

### 输入格式：

每个输入包含 1 个测试用例。每个测试用例第 1 行给出两个正整数 *N*（≤10​4​​，总人数）和 *K*（≤10，总排数）。随后 *N* 行，每行给出一个人的名字（不包含空格、长度不超过 8 个英文字母）和身高（[30, 300] 区间内的整数）。

### 输出格式：

输出拍照的队形。即K排人名，其间以空格分隔，行末不得有多余空格。注意：假设你面对拍照者，后排的人输出在上方，前排输出在下方。

### 输入样例：

10 3

Tom 188

Mike 170

Eva 168

Tim 160

Joe 190

Ann 168

Bob 175

Nick 186

Amy 160

John 159

### 输出样例：

Bob Tom Joe Nick

Ann Mike Eva

Tim Amy John

思路：

模拟题，先排序，然后从高到低可以分奇偶在数组里面填数可以用用一个list，头插尾插交替插入。个人感觉list比较方便

代码：

#include <bits/stdc++.h>

using namespace std;

struct node{

string name;

int h;

};

int cmp(node a,node b)

{

if(a.h==b.h)

return a.name<b.name;

return a.h>b.h;

}

void outhang(int begin,int end,node ren[])

{

list<string>lis;

for(int i=begin,k=1;i<end;i++,k++)

{

if(k%2)

lis.push\_back(ren[i].name);

else

lis.push\_front(ren[i].name);

}

cout<<lis.front();lis.pop\_front();

for(list<string>::iterator it=lis.begin();it!=lis.end();it++)

cout<<' '<<\*it;

cout<<endl;

}

int main()

{

int n,k;

node ren[10005];

cin>>n>>k;k=n/k;

for(int i=0;i<n;i++)

{

cin>>ren[i].name>>ren[i].h;

}

sort(ren,ren+n,cmp);

int t=n%k;

outhang(0,k+t,ren);

for(int i=k+t;i<n;i+=k)

outhang(i,i+k,ren);

return 0;

}

1056 组合数的和(15)

给定 N 个非 0 的个位数字，用其中任意 2 个数字都可以组合成 1 个 2 位的数字。要求所有可能组合出来的 2 位数字的和。例如给定 2、5、8，则可以组合出：25、28、52、58、82、85，它们的和为330。

### 输入格式：

输入在一行中先给出 N（1 < N < 10），随后给出 N 个不同的非 0 个位数字。数字间以空格分隔。

### 输出格式：

输出所有可能组合出来的2位数字的和。

### 输入样例：

3 2 8 5

### 输出样例：

330

思路：

根据排列组合，每个数字在十位被计算了n-1次，各位被计算n-1次，所以最后的和就是每个数字的11倍相加

代码：

#include <iostream>

using namespace std;

int main()

{

int n,sum=0,ans;

cin>>n;

for(int i=0;i<n;i++)

{

cin>>ans;

sum+=ans\*11\*(n-1);

}

cout<<sum;

return 0;

}

1057 数零壹(20)

思路：

水，直接求和，位运算数数就OK了

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

char ch;

int \_0=0,\_1=0,sum=0;

while(~scanf("%c",&ch))

{

if(ch>='a'&&ch<='z') sum+=ch-'a'+1;

if(ch>='A'&&ch<='Z') sum+=ch-'A'+1;

}

while(sum)

{

if(sum&1) \_1++;

else \_0++;

sum>>=1;

}

cout<<\_0<<' '<<\_1<<endl;

return 0;

}

1058 选择题(20)

思路：

模拟题，思路简单过程我写的比较麻烦，第二三组测试数据是全对和全错的情况，附上两组测试数据

批改多选题是比较麻烦的事情，本题就请你写个程序帮助老师批改多选题，并且指出哪道题错的人最多。

### 输入格式：

输入在第一行给出两个正整数 N（≤ 1000）和 M（≤ 100），分别是学生人数和多选题的个数。随后 M 行，每行顺次给出一道题的满分值（不超过 5 的正整数）、选项个数（不少于 2 且不超过 5 的正整数）、正确选项个数（不超过选项个数的正整数）、所有正确选项。注意每题的选项从小写英文字母 a 开始顺次排列。各项间以 1 个空格分隔。最后 N 行，每行给出一个学生的答题情况，其每题答案格式为 (选中的选项个数 选项1 ……)，按题目顺序给出。注意：题目保证学生的答题情况是合法的，即不存在选中的选项数超过实际选项数的情况。

### 输出格式：

按照输入的顺序给出每个学生的得分，每个分数占一行。注意判题时只有选择全部正确才能得到该题的分数。最后一行输出错得最多的题目的错误次数和编号（题目按照输入的顺序从 1 开始编号）。如果有并列，则按编号递增顺序输出。数字间用空格分隔，行首尾不得有多余空格。如果所有题目都没有人错，则在最后一行输出 Too simple。

### 输入样例：

3 4

3 4 2 a c

2 5 1 b

5 3 2 b c

1 5 4 a b d e

(2 a c) (2 b d) (2 a c) (3 a b e)

(2 a c) (1 b) (2 a b) (4 a b d e)

(2 b d) (1 e) (2 b c) (4 a b c d)

### 输出样例：

3

6

5

2 2 3 4

代码：

#include <bits/stdc++.h>

using namespace std;

struct timu{

int id,fen,t,zt,ok;

string daan;

};

int cmp(timu a,timu b)

{

if(a.ok==b.ok)

return a.id<b.id;

return a.ok>b.ok;

}

int main()

{

int n,m,stm[1005]={0};

timu ans;

vector<timu>ve;

cin>>n>>m;

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

{

cin>>ans.fen>>ans.t>>ans.zt;

getchar();

getline(cin,ans.daan);

ans.id=i+1;ans.ok=n;

ve.push\_back(ans);

}

for(int i=0;i<n;i++)

{

string s;

getline(cin,s);

int head=0,till=0;

for(int j=0;j<m;j++,head++,till++)

{

int k=0;

while(s[head]!='(') head++;

while(s[till]!=')') {till++;k++;}

string dn=s.substr(head,k);

if(dn[1]-'0'==ve[j].zt)

{

if(dn.substr(3,ve[j].zt\*2-1)==ve[j].daan)

{

stm[i]+=ve[j].fen;

ve[j].ok--;

}

}

}

}

for(int i=0;i<n;i++)

cout<<stm[i]<<endl;

sort(ve.begin(),ve.end(),cmp);

if(ve[0].ok==0) printf("Too simple\n");

else

{

cout<<ve[0].ok;

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

{

if((ve[i].ok==ve[0].ok))cout<<' '<<ve[i].id;

else break;

}

}

return 0;

}

1059 C语言竞赛(20)

C 语言竞赛是浙江大学计算机学院主持的一个欢乐的竞赛。既然竞赛主旨是为了好玩，颁奖规则也就制定得很滑稽：

* 0、冠军将赢得一份“神秘大奖”（比如很巨大的一本学生研究论文集……）。
* 1、排名为素数的学生将赢得最好的奖品 —— 小黄人玩偶！
* 2、其他人将得到巧克力。

给定比赛的最终排名以及一系列参赛者的 ID，你要给出这些参赛者应该获得的奖品。

输入格式：

输入第一行给出一个正整数 *N*（≤10​4​​），是参赛者人数。随后 *N* 行给出最终排名，每行按排名顺序给出一位参赛者的 ID（4 位数字组成）。接下来给出一个正整数 *K* 以及 *K* 个需要查询的 ID。

输出格式：

对每个要查询的 ID，在一行中输出 ID: 奖品，其中奖品或者是 Mystery Award（神秘大奖）、或者是 Minion（小黄人）、或者是 Chocolate（巧克力）。如果所查 ID 根本不在排名里，打印 Are you kidding?（耍我呢？）。如果该 ID 已经查过了（即奖品已经领过了），打印 ID: Checked（不能多吃多占）。

输入样例：

6

1111

6666

8888

1234

5555

0001

6

8888

0001

1111

2222

8888

2222

输出样例：

8888: Minion

0001: Chocolate

1111: Mystery Award

2222: Are you kidding?

8888: Checked

2222: Are you kidding?

思路：

按照题意来就行了，用一个map标记

代码：

#include <bits/stdc++.h>

using namespace std;

bool isPrime(int num)

{

if (num == 2 || num == 3)

{

return true;

}

if (num % 6 != 1 && num % 6 != 5)

{

return false;

}

for (int i = 5; i <= sqrt(num); i += 6)

{

if (num % i == 0 || num % (i+2) == 0)

{

return false;

}

}

return true;

}

int main()

{

int n,ans;cin>>n;

map<int,int>mp;

for(int i=0;i<n;i++)

{

cin>>ans;

mp[ans]=i+1;

}

cin>>n;

for(int i=0;i<n;i++)

{

cin>>ans;

printf("%04d: ",ans);

if(mp[ans]==1) {printf("Mystery Award\n");mp[ans]=-1;}

else if(mp[ans]==0) printf("Are you kidding?\n");

else if(isPrime(mp[ans])) {printf("Minion\n");mp[ans]=-1;}

else if(mp[ans]==-1) printf("Checked\n");

else {printf("Chocolate\n");mp[ans]=-1;}

}

return 0;

}

1060 爱丁顿数(25)

英国天文学家爱丁顿很喜欢骑车。据说他为了炫耀自己的骑车功力，还定义了一个“爱丁顿数” *E* ，即满足有 *E* 天骑车超过 *E* 英里的最大整数 *E*。据说爱丁顿自己的 *E* 等于87。

现给定某人 *N* 天的骑车距离，请你算出对应的爱丁顿数 *E*（≤*N*）。

### 输入格式：

输入第一行给出一个正整数 *N* (≤10​5​​)，即连续骑车的天数；第二行给出 *N* 个非负整数，代表每天的骑车距离。

### 输出格式：

在一行中给出 *N* 天的爱丁顿数。

### 输入样例：

10

6 7 6 9 3 10 8 2 7 8

### 输出样例：

6

思路：

想明白了这道题就很简单了，代码很短，因为定义是“爱丁顿数”E，即满足有E天骑车超过E英里的最大整数E 所以我们先用天代表数组下标，存每天公里数，然后排降序，从头开始找不满足的情况就是最大天数。

代码：

#include <bits/stdc++.h>

using namespace std;

int a[100005]={0};

int cmp(int a,int b){return a>b;}

int main()

{

int n,e=0;cin>>n;

for(int i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

sort(a,a+n,cmp);

for(int i=0;i<n;i++)

{

if(a[i]>i+1) e=i+1;

else break;

}

cout<<e<<endl;

return 0;

}

1061 判断题(15)

判断题的评判很简单，本题就要求你写个简单的程序帮助老师判题并统计学生们判断题的得分。

### 输入格式：

输入在第一行给出两个不超过 100 的正整数 N 和 M，分别是学生人数和判断题数量。第二行给出 M 个不超过 5 的正整数，是每道题的满分值。第三行给出每道题对应的正确答案，0 代表“非”，1 代表“是”。随后 N 行，每行给出一个学生的解答。数字间均以空格分隔。

### 输出格式：

按照输入的顺序输出每个学生的得分，每个分数占一行。

### 输入样例：

3 6

2 1 3 3 4 5

0 0 1 0 1 1

0 1 1 0 0 1

1 0 1 0 1 0

1 1 0 0 1 1

### 输出样例：

13

11

12

思路：

水

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,m,da[105],f[105];

cin>>n>>m;

for(int i=0;i<m;i++) cin>>f[i];

for(int i=0;i<m;i++) cin>>da[i];

for(int i=0;i<n;i++)

{

int ans,sum=0;

for(int j=0;j<m;j++)

{

cin>>ans;

if(ans==da[j]) sum+=f[j];

}

cout<<sum<<endl;

}

return 0;

}

1062 最简分数(20)

一个分数一般写成两个整数相除的形式：*N*/*M*，其中 *M* 不为0。最简分数是指分子和分母没有公约数的分数表示形式。

现给定两个不相等的正分数 *N*​1​​/*M*​1​​ 和 *N*​2​​/*M*​2​​，要求你按从小到大的顺序列出它们之间分母为 *K* 的最简分数。

### 输入格式：

输入在一行中按 *N*/*M* 的格式给出两个正分数，随后是一个正整数分母 *K*，其间以空格分隔。题目保证给出的所有整数都不超过 1000。

### 输出格式：

在一行中按 *N*/*M* 的格式列出两个给定分数之间分母为 *K* 的所有最简分数，按从小到大的顺序，其间以 1 个空格分隔。行首尾不得有多余空格。题目保证至少有 1 个输出。

### 输入样例：

7/18 13/20 12

### 输出样例：

5/12 7/12

思路：

就是化简分数问题，把输入的两个分数之间的以K为底的分数遍历一遍找出不能化简的输出。

代码：

#include <bits/stdc++.h>

using namespace std;

int flag=0;

int gcd(int a,int b)

{

return b?gcd(b,a%b):a;

}

double getf()

{

double a,b;

scanf("%lf/%lf",&a,&b);

return a/b;

}

int out\_huajian(int fz,int fm)

{

int g=gcd(fz,fm);

if(g==1)

{

if(flag) printf(" ");

printf("%d/%d",fz/g,fm/g);

flag=1;

}

}

int main()

{

double a,b;

int k;

a=getf();b=getf();

if(a>b) swap(a,b);

cin>>k;

for(double i=0;i/k<b;i++)

{

if(i/k>a) out\_huajian(i,k);

}

return 0;

}

1063 计算谱半径(20)

在数学中，矩阵的“谱半径”是指其特征值的模集合的上确界。换言之，对于给定的 *n* 个复数空间的特征值 { *a*​1​​+*b*​1​​*i*,⋯,*a*​*n*​​+*b*​*n*​​*i* }，它们的模为实部与虚部的平方和的开方，而“谱半径”就是最大模。

现在给定一些复数空间的特征值，请你计算并输出这些特征值的谱半径。

### 输入格式：

输入第一行给出正整数 N（≤ 10 000）是输入的特征值的个数。随后 N 行，每行给出 1 个特征值的实部和虚部，其间以空格分隔。注意：题目保证实部和虚部均为绝对值不超过 1000 的整数。

### 输出格式：

在一行中输出谱半径，四舍五入保留小数点后 2 位。

### 输入样例：

5

0 1

2 0

-1 0

3 3

0 -3

### 输出样例：

4.24

思路：

简单计算求最大值

代码：

#include <bits/stdc++.h>

using namespace std;

double p(int a,int b)

{

return sqrt(a\*a+b\*b);

}

int main()

{

int n,a,b;

double maxx=0;

cin>>n;

while(n--)

{

cin>>a>>b;

maxx=max(p(a,b),maxx);

}

printf("%.2lf\n",maxx);

return 0;

}

1064 朋友数(20)

如果两个整数各位数字的和是一样的，则被称为是“朋友数”，而那个公共的和就是它们的“朋友证号”。例如 123 和 51 就是朋友数，因为 1+2+3 = 5+1 = 6，而 6 就是它们的朋友证号。给定一些整数，要求你统计一下它们中有多少个不同的朋友证号。

### 输入格式：

输入第一行给出正整数 N。随后一行给出 N 个正整数，数字间以空格分隔。题目保证所有数字小于 10​4​​。

### 输出格式：

首先第一行输出给定数字中不同的朋友证号的个数；随后一行按递增顺序输出这些朋友证号，数字间隔一个空格，且行末不得有多余空格。

### 输入样例：

8

123 899 51 998 27 33 36 12

### 输出样例：

4

3 6 9 26

思路：

把计算的朋友数都放进set里面，最后输出set中元素的数量和元素就好了

代码：

#include <bits/stdc++.h>

using namespace std;

int pys(int n)

{

if(n) return n%10+pys(n/10);

return 0;

}

int main()

{

int n,ans,flag=0;

set<int>se;

cin>>n;

for(int i=0;i<n;i++)

{

cin>>ans;

se.insert(pys(ans));

}

cout<<se.size()<<endl;

set<int>::iterator it=se.begin();

cout<<\*it;it++;

for(;it!=se.end();it++)

cout<<' '<<\*it;

return 0;

}

1065 单身狗(25)

单身狗”是中文对于单身人士的一种爱称。本题请你从上万人的大型派对中找出落单的客人，以便给予特殊关爱。

### 输入格式：

输入第一行给出一个正整数 N（≤ 50 000），是已知夫妻/伴侣的对数；随后 N 行，每行给出一对夫妻/伴侣——为方便起见，每人对应一个 ID 号，为 5 位数字（从 00000 到 99999），ID 间以空格分隔；之后给出一个正整数 M（≤ 10 000），为参加派对的总人数；随后一行给出这 M 位客人的 ID，以空格分隔。题目保证无人重婚或脚踩两条船。

### 输出格式：

首先第一行输出落单客人的总人数；随后第二行按 ID 递增顺序列出落单的客人。ID 间用 1 个空格分隔，行的首尾不得有多余空格。

### 输入样例：

3

11111 22222

33333 44444

55555 66666

7

55555 44444 10000 88888 22222 11111 23333

### 输出样例：

5

10000 23333 44444 55555 88888

思路：

水题，记录成对的人，然后在后面的人中遍历如果他的对象不在或没有对象就存下来，最后输出。

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a,b,n,flag=0;

map<int,int>mp;

vector<int>ve,dog;

cin>>n;

for(int i=0;i<n;i++)

{

cin>>a>>b;

mp[a]=b;mp[b]=a;

}

cin>>n;

for(int i=0;i<n;i++)

{

int ans;cin>>ans;

ve.push\_back(ans);

}

sort(ve.begin(),ve.end());

for(int i=0;i<n;i++)

{

if(!binary\_search(ve.begin(),ve.end(),mp[ve[i]]))

{

dog.push\_back(ve[i]);

}

}

cout<<dog.size()<<endl;

for(int i=0;i<dog.size();i++)

{

printf("%05d%c",dog[i],i==dog.size()-1?'\n':' ');

}

return 0;

}

1066 图像过滤(15)

图像过滤是把图像中不重要的像素都染成背景色，使得重要部分被凸显出来。现给定一幅黑白图像，要求你将灰度值位于某指定区间内的所有像素颜色都用一种指定的颜色替换。

### 输入格式：

输入在第一行给出一幅图像的分辨率，即两个正整数 *M* 和 *N*（0<*M*,*N*≤500），另外是待过滤的灰度值区间端点 *A* 和 *B*（0≤*A*<*B*≤255）、以及指定的替换灰度值。随后 *M* 行，每行给出 *N* 个像素点的灰度值，其间以空格分隔。所有灰度值都在 [0, 255] 区间内。

### 输出格式：

输出按要求过滤后的图像。即输出 *M* 行，每行 *N* 个像素灰度值，每个灰度值占 3 位（例如黑色要显示为 000），其间以一个空格分隔。行首尾不得有多余空格。

### 输入样例：

3 5 100 150 0

3 189 254 101 119

150 233 151 99 100

88 123 149 0 255

### 输出样例：

003 189 254 000 000

000 233 151 099 000

088 000 000 000 255

思路：

水题

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,m,mn[505][505],a,b,k;

cin>>n>>m>>a>>b>>k;

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

cin>>mn[i][j];

if(mn[i][j]<=b&&mn[i][j]>=a) mn[i][j]=k;

}

}

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

printf("%03d%c",mn[i][j],j==m-1?'\n':' ');

}

}

return 0;

}

1067 试密码(20)

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。

### 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

思路：

字符串处理的题目，注意几个坑点

+ 正确的密码是没有空格，但是错误密码不一定，所以要读取一整行

+ 整行只有一个“#”才是结束，不要至判断一个s[0]==‘#’就结束

+ 如果错误次数达到N，要先输出”Wrong password: XXXXXXX”，再输出”Account locked”

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

string pw,s;

int n;

cin>>pw>>n;getchar();

while(n--)

{

getline(cin,s);

if(s=="#") return 0;

if(s==pw)

{

cout<<"Welcome in"<<endl;

return 0;

}

else cout<<"Wrong password: "<<s<<endl;

}

cout<<"Account locked"<<endl;

return 0;

}

1068 万绿丛中一点红(20)

对于计算机而言，颜色不过是像素点对应的一个 24 位的数值。现给定一幅分辨率为 *M*×*N* 的画，要求你找出万绿丛中的一点红，即有独一无二颜色的那个像素点，并且该点的颜色与其周围 8 个相邻像素的颜色差充分大。

### 输入格式：

输入第一行给出三个正整数，分别是 *M* 和 *N*（≤ 1000），即图像的分辨率；以及 TOL，是所求像素点与相邻点的颜色差阈值，色差超过 TOL 的点才被考虑。随后 *N* 行，每行给出 *M* 个像素的颜色值，范围在 [0,2​24​​)内。所有同行数字间用空格或 TAB 分开。

### 输出格式：

在一行中按照 (x, y): color 的格式输出所求像素点的位置以及颜色值，其中位置 x 和 y 分别是该像素在图像矩阵中的列、行编号（从 1 开始编号）。如果这样的点不唯一，则输出 Not Unique；如果这样的点不存在，则输出 Not Exist。

### 输入样例 1：

8 6 200

0 0 0 0 0 0 0 0

65280 65280 65280 16711479 65280 65280 65280 65280

16711479 65280 65280 65280 16711680 65280 65280 65280

65280 65280 65280 65280 65280 65280 165280 165280

65280 65280 16777015 65280 65280 165280 65480 165280

16777215 16777215 16777215 16777215 16777215 16777215 16777215 16777215

### 输出样例 1：

(5, 3): 16711680

### 输入样例 2：

4 5 2

0 0 0 0

0 0 3 0

0 0 0 0

0 5 0 0

0 0 0 0

### 输出样例 2：

Not Unique

### 输入样例 3：

3 3 5

1 2 3

3 4 5

5 6 7

### 输出样例 3：

Not Exist

思路：

这是乙级里面正确率最低的一道题目，确实比较坑。坑点

+ 说的是周围八个，但是不能从内圈开始，外圈的虽然不够八个但是也要算上周围有几个是几个

+ 这个点的值要是唯一的，因为这个刚开始样例都看不明白

+ 注意n,m是先列再行，输出的时候也是。

代码：

#include <bits/stdc++.h>

using namespace std;

int mn[1002][1002];

int dir[8][2]={-1,1,0,1,1,1,-1,0,1,0,-1,-1,0,-1,1,-1};

int main()

{

int n,m,TOL,X,Y,color,flag=1;

map<int,int>mp;

cin>>m>>n>>TOL;

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

scanf("%d",&mn[i][j]);

mp[mn[i][j]]++;

}

}

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

int TolOk=1;

if(mp[mn[i][j]]>1) continue;

for(int k=0;k<8;k++)

{

int x=i+dir[k][0],y=j+dir[k][1];

if(x>=0&&x<n&&y>=0&&y<m)

{

if(abs(mn[x][y]-mn[i][j])<=TOL) {TolOk=0;break;}

}

}

if(TolOk)

{

if(flag)

{

X=i+1;Y=j+1;color=mn[i][j];

flag=0;

}

else

{

cout<<"Not Unique"<<endl;

return 0;

}

}

}

}

if(flag) cout<<"Not Exist"<<endl;

else cout<<'('<<Y<<", "<<X<<"): "<<color<<endl;

return 0;

}

1069 微博转发抽奖(20)

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...。

### 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...

思路：

从M位开始找，每k个判断是否中过奖循环判断（判断时要注意越界）

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

map<string,int>mp;

vector<string>ve;

int n,k,m,flag=1;

cin>>n>>k>>m;

m--;

for(int i=0;i<n;i++)

{

string s;cin>>s;

ve.push\_back(s);

}

for(int i=m,t=0;i<n;i++,t++)

{

if(i==m||t==k)

{

for(;mp[ve[i]]&&i<n;i++);

if(i==n) break;

mp[ve[i]]=1;

cout<<ve[i]<<endl;

t=0;flag=0;

}

}

if(flag) cout<<"Keep going..."<<endl;

return 0;

}

1070 结绳（贪心）(25)

给定一段一段的绳子，你需要把它们串成一条绳。每次串连的时候，是把两段绳子对折，再如下图所示套接在一起。这样得到的绳子又被当成是另一段绳子，可以再次对折去跟另一段绳子串连。每次串连后，原来两段绳子的长度就会减半。



给定 *N* 段绳子的长度，你需要找出它们能串成的绳子的最大长度。

### 输入格式：

每个输入包含 1 个测试用例。每个测试用例第 1 行给出正整数 *N* (2≤*N*≤10​4​​)；第 2 行给出 *N* 个正整数，即原始绳段的长度，数字间以空格分隔。所有整数都不超过10​4​​。

### 输出格式：

在一行中输出能够串成的绳子的最大长度。结果向下取整，即取为不超过最大长度的最近整数。

### 输入样例：

8

10 15 12 3 4 13 1 15

### 输出样例：

14

思路：

简单贪心，因为每次绳子都要对折所以从短的开始对折这样可以最大程度减少长绳子的损失。

代码：

#include <bits/stdc++.h>

using namespace std;

int main()

{

double sum;

int a[10004],n;

cin>>n;

for(int i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

sort(a,a+n);

sum=a[0];

for(int i=1;i<n;i++)

{

sum+=a[i];

sum/=2;

}

printf("%d\n",(int)sum);

return 0;

}