|  |  |
| --- | --- |
| Problem B | 七龙珠 |

## **Description**

就是上面分别有一二三四五六七颗星的龙珠

## **Input**

第一行有一个整数n，代表输入实例有n(题目保证n<=100)组。

接下来的n行，每行为一个字符串k（题目保证k的长度<=3000），

字符串中每出现一个连续的“one”，代表一星龙珠

同理，“two”代表二星龙珠

“three”代表三星龙珠

“four”代表四星龙珠

“five”代表五星龙珠

“six”代表六星龙珠

“seven”代表七星龙珠

同样的龙珠可能找到不止一颗。

输入保证字符串中的字符全为小写字母。

字母可以出现连续。

如onetwo可以视为找到1,2两颗龙珠，twone和twoone都可以视为找到1,2两颗龙珠(因为他们都包含连续的one/two)。中间可能夹杂着无关字母，比如oneptwo，因为包含one/two，所以也可以被视为1,2两个龙珠。

## **Output**

对于每组输入实例，请输出七种龙珠是否已经找齐。如果是，请输出“Yes”，否则请输出“No”。

## **Sample Input**

3  
twonethreefourfivesixseven  
oneatwobthreecfourdfiveesixfsevensevenseveneightnine  
threedaysthreenights

## **Sample Output**

Yes  
Yes  
No

最笨的方法 逐个访问

#include<stdio.h>

#include<string.h>

using namespace std;

int main()

{

int t;

scanf("%d\n",&t);

while(t--)

{

char a[10005];

int i,c=0;

gets(a);

int len=strlen(a);

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

{

if(a[i]=='o'&&a[i+1]=='n'&&a[i+2]=='e')

c++;

}

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

{

if(a[i]=='t'&&a[i+1]=='w'&&a[i+2]=='o')

c++;

}

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

{

if(a[i]=='t'&&a[i+1]=='h'&&a[i+2]=='r'&&a[i+3]=='e'&&a[i+4]=='e')

c++;

}

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

{

if(a[i]=='f'&&a[i+1]=='o'&&a[i+2]=='u'&&a[i+3]=='r')

c++;

}

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

{

if(a[i]=='f'&&a[i+1]=='i'&&a[i+2]=='v'&&a[i+3]=='e')

c++;

}

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

{

if(a[i]=='s'&&a[i+1]=='i'&&a[i+2]=='x')

c++;

}

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

{

if(a[i]=='s'&&a[i+1]=='e'&&a[i+2]=='v'&&a[i+3]=='e'&&a[i+4]=='n')

c++;

}

if(c>=7)

printf("Yes\n");

else

printf("No\n");

}

return 0;

}

利用函数查找：

#include <stdio.h>

#include <string.h>

**int** myfunc();

**int** main()

{

**int** n;

**scanf**("%d", &n);

**while** (n--)

**if** (myfunc())

**printf**("Yes\n");

**else**

**printf**("No\n");

**return** 0;

}

**int** myfunc(){

**char** str[3100];

**char** \*p;

**memset**(str, 0, 3100);

**scanf**("%s", str);

p = **strstr**(str, "one");

**if** (p == 0) **return** 0;

p = **strstr**(str, "two");

**if** (p == 0) **return** 0;

p = **strstr**(str, "three");

**if** (p == 0) **return** 0;

p = **strstr**(str, "four");

**if** (p == 0) **return** 0;

p = **strstr**(str, "five");

**if** (p == 0) **return** 0;

p = **strstr**(str, "six");

**if** (p == 0) **return** 0;

p = **strstr**(str, "seven");

**if** (p == 0) **return** 0;

**return** 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem: 1511

User: 201611106021

Language: C

Result: Accepted

Time:0 ms

Memory:948 kb

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

|  |  |
| --- | --- |
| Problem C | 地精和侏儒 |

简单的排序后输出第一个和最后一个值

## **Description**

地精和侏儒

## **Input**

第一行为T（保证T<1000)，表示有T组数据。

之后的T组，每组开头为数列中数的个数N（保证3<N<50)。

之后的N个数，是数列中的数字。

## **Output**

针对每一组输入，在单独的一行中分别输出最大的数和最小的数。

## **Sample Input**

3  
3  
9 5 10  
4  
1 3 5 7  
5  
8 8 10 10 2

## **Sample Output**

10 5  
7 1  
10 2

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

int main()

{

int t;

scanf("%d\n",&t);

while(t--)

{

int n,a[10005];

scanf("%d",&n);

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

{

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

}

sort(a,a+n);

printf("%d %d\n",a[n-1],a[0]);

}

return 0;

}

不排序解法：

#include <stdio.h>

**int** main(){

**int** t, n, i, num, max, min;

**scanf**("%d", &t);

**while** (t--){

**scanf**("%d", &n);

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

**scanf**("%d", &num);

**if** (i == 0)

max = min = num;

**else**{

**if** (num > max) max = num;

**if** (num < min) min = num;

}

}

**printf**("%d %d\n", max, min);

}

**return** 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem: 1512

User: 201611106021

Language: C

Result: Accepted

Time:0 ms

Memory:944 kb

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

|  |  |
| --- | --- |
| Problem E | 你画我猜 |

每个数猜对和猜错的概率是一样的，也就是说那人不长记性，然后就是数学问题了

## **Description**

现在就请你来计算一下这个概率，如果词库中有x个词，这x个词你全都记住了，你在完全看不懂画的人画的是什么的情况下，每次随机从这x个词中猜一个词，一共猜了m次，请问你有多大的机率可以猜中呢？

“这其实是个数学题！”小兰说道。

## **Input**

第一行有一个整数n，代表输入实例有n组。接下来的n行，每行两个整数x(1<=x<=10000)，m(1<=m<=100)，分别代表词库中的词数和你猜的次数。

## **Output**

对于每组输入实例，请输出你猜中的概率，结果保留4位小数。

## **Sample Input**

2  
3 1  
2 2

## **Sample Output**

0.3333  
0.7500

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

using namespace std;

int main()

{

int t;

scanf("%d\n",&t);

while(t--)

{

double m,n,sum=0,c=1;

scanf("%lf%lf",&m,&n);

double e;

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

{

if(c==1)

c=1/m;

else{

e=1/m;

c=e\*pow((1-e),i);}

sum+=c;

}

printf("%.4f\n",sum);

}

return 0;

}

|  |  |
| --- | --- |
| Problem F | 不服sala续集 |

这个就是个数组然后求平均值，没啥可说的

## **Description**

## 出小明的战斗力和对面5个人的战斗力，请输出小明在这一局游戏中是自信满满还是慌得要死。

## **Input**

第一行T，表示有T组数据（保证T小于1000）。

每组数据中的第一行，是一个整数或浮点数，为小明的战斗力。

第二行中有五个整数或浮点数，表示对手每个人的战斗力。

## **Output**

对每一组数据，如果小明的战斗力大于等于对面战斗力的平均值，输出” Xiaoming is full of confidence”，反之输出” Xiaoming is panic to death”

## **Sample Input**

3  
1000  
1000 2000 3000 4000 5000  
2500  
2000 2000 2000 2000 2000  
2500  
1000 3000 1500 2500 2500

## **Sample Output**

Xiaoming is panic to death  
Xiaoming is full of confidence  
Xiaoming is full of confidence

#include<stdio.h>

using namespace std;

int main()

{

int t;

scanf("%d",&t);

while(t--)

{

double n,sum=0;

scanf("%lf",&n);

double a[5];

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

{

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

sum+=a[i];

}

if(n>=sum/5)

printf("Xiaoming is full of confidence\n");

else

printf("Xiaoming is panic to death\n");

}

return 0;

}

|  |  |
| --- | --- |
| Problem G | 开荒专用闹钟 |

声明两个数组后他的下标，表示时间，内容表示是否是闹铃，对设置的闹钟限制条件，然后逐个访问如果数组里的值不为零且符合小时和分钟的条件就输出。

## **Description**

他按照以下三种规则开始设置闹铃（定个闹铃还按照规则，有病吧）：

1． 从00：00开始，每15分钟设置一个闹铃（这要设置多少个，用个for循环一次设置完可不可以）。

2． 在分钟以8为尾数时，设置闹铃。（某个人喜欢8这个数字，所以这条规则是可以原谅的对吧）。

3． 每当小时数加1可以被分钟数加1整除的时候，设置闹铃。比如03：01，3+1=4，1+1=2，4可以被2整除。

不过他还是不放心，我定的第x个闹铃到底应该什么时候响来着？

## **Input**

第一行有一个整数n，代表输入实例有n组。接下来的n行，每行一个整数x(保证1<=x<=80)，代表第x个闹铃。

## **Output**

对于每个整数x，请输出x对应闹铃的时间，时间格式用hh：mm表示。

时钟的时和分始终要保持两位（C语言用printf(“%02d”)，C++语言用cout<<setw(2)<<setfill(‘0’)，具体用法请看你自己带进来的资料。什么你没准备？请吸取教训充分准备。）

## **Sample Input**

3  
1  
3  
9

## **Sample Output**

00:00  
00:15  
00:48

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

using namespace std;

int main()

{

int t;

scanf("%d",&t);

while(t--)

{

int a[60]={0},b[60]={0};

for(int i=0;i<60;i+=15)

a[i]=1;

for(int i=0;i<60;i++)

{

if(i%10==8)

a[i]=1;

}

int r,c=0;

scanf("%d",&r);

for(int i=0;i<60;i++)

{

for(int j=0;j<60;j++)

{

if(a[j]==1||(i+1)%(j+1)==0)

c++;

if(c==r){

printf("%02d:%02d\n",i,j);

break;}

}

}

}

return 0;

}

我的解法：

#include <stdio.h>

#include<math.h>

**int** main()

{

**int** t;

**scanf**("%d",&t);

**while**(t--){

**int** a;

**scanf**("%d",&a);

**int** h=0,m=0;

**while**(a!=0){

**if**(m%15==0){

a--;

}

**else** **if**(m%10==8){

a--;

}

**else** **if**((h+1)%(m+1)==0){

a--;

}

m++;

**if**(m==60){h++,m=0;}

}

**printf**("%02d:%02d\n",h,m-1);

}

}

|  |  |
| --- | --- |
| Problem H | 使命召唤 |

这个题就是按照那个比例进行放大，然后只要不超过那个屏幕的大小就好了

## **Description**

但是他的屏幕分辨率有点奇怪，总是达不到标准的分辨率比例。比如现在流行的是16:9，可是他屏幕的分辨率经常是1920\*1200这种非标准版本的，会导致游戏无法正常运行。为了适应16:9的比例，他会手动调整到1920\*1080，虽然这样会导致屏幕有黑边，但是起码保持了比例（射击的时候不容易误判），而且这样黑边最小，可视面积最大。

现在问题来了。给出小明当前屏幕的分辨率和要求的比例，输出他能够手动调整的最大分辨率。如果没有办法调整……比如分辨率是1\*1，但是游戏要求的比例是1:2（当然，只是举个例子），他就没有办法调整了，这个时候，输出0 0.

（原题目具体的坑我给忘了，这个是简化版的）

## **Input**

第一行为组数T（保证1<T<50）。

对于每组数据，在一行中输入4个数字，前两个数字代表当前的分辨率，后两个数字代表屏幕要求的分辨率比例。

## **Output**

在单独的一行中输出小明能够手动调整的最大分辨率。如果有多种方案，输出分辨率面积最大的1种，如果没有方案，输出0 0

## **Sample Input**

3  
800 600 4 3  
1920 1200 16 9  
1 1 1 2

## **Sample Output**

800 600  
1920 1080  
0 0

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

using namespace std;

int main()

{

int t;

scanf("%d\n",&t);

while(t--)

{

double a,b,c,d,sum=0;

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

double m=a/c;

double n=b/d;

if(a<c||b<d)

printf("0 0\n");

else

{

if(m>=n)

printf("%.0f %.0f\n",c\*n,b);

else

printf("%.0f %.0f\n",a,d\*m);

}

}

return 0;

}

Ps.若比例为1600:900，和16:9，此时结果应该一致，题目实际上被简化了很多。

|  |  |
| --- | --- |
| Problem I | 不服sala第三部 |

数学问题，题目中很显然只要给出样例就一定能判断输赢，这个关键看第一个人说几，每个人都会增1-m之间的数，也就是说无论对方说几我都可以凑m+1，这样只要x能被m+1整除那么那个人说0的人一定能赢

## **Description**

先由小红数一个0，之后两人交替数数，每次在前一个人数的基础上往后数一个数或两个数，谁能数到数字30谁就获胜。

在经过几次游戏之后，小明发现他永远赢不了游戏（你的计谋呢，你的担架呢？），数学课代表的力量太强大了，怪不得小涛找她当闺蜜- -。于是他们决定改变一下游戏的规则。

他们随机选定两个数字m和x，还是由小红先数一个数0，之后两人交替数数，每次可以在前一个人数的基础上往后数1到m个数，谁先数到数x谁就获胜。

现在已知两人都绝顶聪明，请问谁有必胜的策略？

## **Input**

第一行有一个整数n，代表输入实例有n组。接下来的n行，每行两个整数m(1<=m<=100)，x(1<=x<=10000)，如题目所述。

## **Output**

对于每组输入实例，请输出谁有必胜的策略，如果小明必胜，请输出“Xiaoming”，如果小红必胜，请输出“Xiaohong”。

## **Sample Input**

3  
2 30  
1 1  
13 14

## **Sample Output**

Xiaohong  
Xiaoming  
Xiaohong

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

using namespace std;

int main()

{

int t;

scanf("%d",&t);

while(t--)

{

int m,n;

scanf("%d %d",&m,&n);

if(n%(m+1)==0)

{

printf("Xiaohong\n");

}

else

printf("Xiaoming\n");

}

return 0;

}

|  |  |
| --- | --- |
| Problem J | 炉石秘籍 |

数字只有1，2，3都是一位数也就是说每隔一个字符+便是一个数字，所以可以把数字都存到一个数组里排序，然后输出（输出和以往的输出一样只不过把我们熟悉的空格变成了+）

## **Description**

小涛有个毛病，他只能从小到大加不能从大到小加（学名：他只能进行非降序的运算）。比如计算3+2+1，他必须把顺序换一换变成1+2+3。计算1+3+1+3+1，他只能先转换成1+1+1+3+3才能够进行计算。

现在给出一系列加法，你能否将这个加法式子转换成小涛能够看懂的顺序？

## **Input**

第一行是组数T（保证1<T<50）。

每组数据包含一个字符串(字符串保证长度小于100)，这个字符串只包含1,2,3和加号（1个数的话，也有可能没有加号），组成一个加法式子。

## **Output**

在单独的一行中输出小涛可以计算的排序以后的式子。

## **Sample Input**

3  
3+2+1  
1+1+3+1+3  
2

## **Sample Output**

1+2+3  
1+1+1+3+3  
2

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

using namespace std;

int main()

{

int t;

scanf("%d\n",&t);

while(t--)

{

char a[10005],b[10005];

gets(a);

int j=0,len=strlen(a);

for(int i=0;i<len;i+=2)

b[j++]=a[i];

sort(b,b+j);

for(int i=0;i<j-1;i++)

printf("%c+",b[i]);

printf("%c\n",b[j-1]);

}

return 0;

}

## **Description**

## 假设他们每个人都会m个英雄，每个英雄的战斗力是固定的，战斗力高的英雄必定能够战胜战斗力低的英雄（比如小明虽然只会诺克萨斯之手，但是他诺手的战斗力有10000，打小涛战斗力9999的剑圣也能每次都获胜），战斗力相同的话算平局，双方都不用给Q币。 小明因为比较弱，可以自由的指定自己英雄的出场顺序和小涛英雄的出场顺序（怎么感觉像某战国时期的故事呢），小涛不能拒绝。双方每个英雄都只能出场1次，且必须出场。

现在问题来了，在小明选取最优策略的情况下，他最多可以赚小涛多少QB呢？要是小明无论如何都赢不了小涛，那么输出小涛能赢小明多少QB

## **Input**

第一行T，输入的组数。

对每一组数据，第一行m，为他们会的英雄的个数（1<m<10），之后的一行有m个数据(题目保证m小于100)，代表小明会的m个英雄的战斗力。接下来的1行也有m个数据，代表小涛会的m个英雄的战斗力。

## **Output**

对每组输入的数据，在单独的一行中输出。

如果小明在选择最优对阵情况下可以从小涛那里获得Q币，比如小明有三个英雄，战斗力1000,2000,3000，小涛有三个英雄战斗力1500,2500,3500，小明的最优策略是用1000对3500（输200），3000vs2500（赢200），2000vs1500（赢200），总共获得200QB，那么输出”Xiaoming will get 200QB”，如果双方打平的话输出”Xiaoming will get 0QB”

如果小明采用最优策略也无法赢小涛Q币，比如小明两个英雄战斗力1000,1000，小涛战斗力18000,20000，两局都会输，那小明怎么努力也打不过小涛（青铜五用战术打最强王者是没用的），净输400QB，那么输出”Xiaotao will get 400QB”。

## **Sample Input**

4  
3  
9200 8300 7100  
9500 8700 7400  
2  
2000 2000  
2000 2000  
2  
2000 1900  
2200 1800  
2  
1800 1700  
1900 2000

## **Sample Output**

Xiaoming will get 200QB  
Xiaoming will get 0QB  
Xiaoming will get 0QB  
Xiaotao will get 400QB

http://blog.csdn.net/lawrencesgj/article/details/8001638