目录

[一、训练记录 2](#_Toc461699771)

[（一）Training Day 1 2](#_Toc461699772)

[（二）Training Day 2 12](#_Toc461699773)

[（三）Training Day 3 17](#_Toc461699774)

[（四）Training Day 4 28](#_Toc461699775)

[（五）Training Day 5 37](#_Toc461699776)

[（六）Training Day 6 40](#_Toc461699777)

[（七）Training Day 7 49](#_Toc461699778)

[（九）Training Day 9 74](#_Toc461699779)

# 一、训练记录

## （一）Training Day 1

A.对于一群NP（2<=NP<=10）个要互送礼物的朋友，郭铮鹏要确定每个人送出的钱比收到的多多少。 在这一个问题中，每个人都准备了一些钱来送礼物，而这些钱将会被平均分给那些将收到他的礼物的人。 然而，在任何一群朋友中，有些人将送出较多的礼物(可能是因为有较多的朋友)，有些人有准备了较多的钱。 给出一群朋友，没有人的名字会长于 14 字符，给出每个人将花在送礼上的钱，和将收到他的礼物的人的列表， 请确定每个人收到的比送出的钱多的数目。

输入格式：

第 1 行: 人数NP,2<= NP<=10

第 2 行 到 第NP+1 行:这NP个在组里人的名字　一个名字一行

第NP＋2到最后：

这里的I段内容是这样组织的：

第一行是将会送出礼物人的名字。

第二行包含二个数字:　第一个是送出的钱的数目（在0到2000的范围里），第二个 NGi 是将收到这个人礼物的人的个数 如果 NGi 是非零的, 在下面 NGi 行列出礼物的接受者的名字，一个名字一行。

输出格式：

输出 NP 行

每行是一个的名字加上空格再加上收到的比送出的钱多的数目。

对于每一个人，他名字的打印顺序应和他在输入的2到NP＋1行中输入的顺序相同。所有的送礼的钱都是整数。

每个人把相同数目的钱给每位要接受礼物的朋友，而且尽可能多给，不能给出的钱（即无法被整除的钱）由送礼者本人持有。

样例输入

5

dave

laura

owen

vick

amr

dave

200 3

laura

owen

vick

owen

500 1

dave

amr

150 2

vick

owen

laura

0 2

amr

vick

vick

0 0

样例输出

dave 302

laura 66

owen -359

vick 141

amr -150

代码：

#include<iostream>

#include<string.h>

#include<algorithm>

using namespace std;

const int maxn=15;

int order[maxn];

int o=0;

struct per{

char name[15];

int out;

int in;

int sum;

}person[maxn];

int main(){

int NP;

scanf("%d",&NP);

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

scanf("%s",&person[i].name);

person[i].in=0;

person[i].out=0;

person[i].sum=0;

}

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

int NGi;

char NAME[15];

scanf("%s",NAME);

for(int r=0;r<NP;r++){ //输入人名并判断

if(strcmp(person[r].name,NAME)==0){

order[o]=r;

o++;

break;

}

}

scanf("%d %d",&person[order[o-1]].out,&NGi);

char receiver[15];

if (NGi!=0)

{ int j=person[order[o-1]].out%NGi;

person[order[o-1]].out=(j==0)?person[order[o-1]].out:person[order[o-1]].out-j;

//printf("%d\n",person[order[o-1]].out);

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

{scanf("%s",receiver);

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

{if(strcmp(person[k].name,receiver)==0)

{person[k].in+=person[order[o-1]].out/NGi;

//printf("%d\n",person[k].in);

break;

}

}

}

}

//if(NGi!=0){

}

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

{person[i].sum=person[i].in-person[i].out;

}

sort(order,order+NP);

for(o=0;o<NP-1;o++){

printf("%s %d\n",person[order[o]].name,person[order[o]].sum);

}

printf("%s %d",person[order[NP-1]].name,person[order[NP-1]].sum);

return 0;

}

B. 郭铮鹏印象中有好多个13号是星期五，13号在星期五比在其他日子少吗?为了回答这个问题,写一个程序，要求计算每个月的十三号落在周一到周日的次数。给出N年的一个周期，要求计算1900年1月1日至1900+N-1年12月31日中十三号落在周一到周日的次数，用现成的函数

请不要预先算好数据（就是叫不准打表）!

输入格式：

一个正整数n

输出格式：

七个在一行且相分开的整数,它们代表13日是星期六,星期日,星期一...星期五的次数..

样例输入

20

样例输出

36 33 34 33 35 35 34

代码：

#include<iostream>

using namespace std;

int mon[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

int day[7] = { 0, 0, 0, 0, 0, 0, 0 };

int main(){

int k = 5, n;

cin >> n;

n += 1900;

for (int i = 1900; i<n; i++){

if ((i % 100 != 0 && i % 4 == 0) || i % 400 == 0)

mon[1] = 29;

for (int j = 0; j<12; j++){

day[k]++;

k += mon[j];

k %= 7;

}

mon[1] = 28;

}

for (int i = 5; i<7; i++)

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

for (int i = 0; i<4; i++)

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

printf("%d",day[4]);

return 0;

}

C. 郭铮鹏有一条由N个红色的，白色的，或蓝色的珠子组成的项链(3<=N<=350)，珠子是随意安排的。 这里是 n=29 的二个例子:

1 2 1 2

r b b r b r r b

r b b b

r r b r

r r w r

b r w w

b b r r

b b b b

b b r b

r r b r

b r r r

b r r r

r r r b

r b r r r w

图片 A 图片 B

r 代表 红色的珠子

b 代表 蓝色的珠子

w 代表 白色的珠子

第一和第二个珠子在图片中已经被作记号。

图片 A 中的项链可以用下面的字符串表示：

brbrrrbbbrrrrrbrrbbrbbbbrrrrb

郭铮鹏要在一些点打破项链,展开成一条直线，然后从一端开始收集同颜色的珠子直到遇到一个不同的颜色珠子，在另一端做同样的事(颜色可能与在这之前收集的不同)。他想让你帮忙确定应该在哪里打破项链来收集到最大数目的珠子。

例如，在图片 A 中的项链中，在珠子 9 和珠子 10 或珠子 24 和珠子 25 之间打断项链可以收集到8个珠子。

白色珠子是特殊的：

在一些项链中还包括白色的珠子(如图片B) 所示。

当收集珠子的时候，一个被遇到的白色珠子可以被当做红色也可以被当做蓝色且被涂上这种颜色。

表现含有白珠项链的字符串将会包括三个符号 r ， b 和 w 。

写一个程序来确定从一条被给出的项链可以收集到的珠子最大数目。

输入格式：

第 1 行: N, 珠子的数目

第 2 行: 一串长度为N的字符串, 每个字符是 r ， b 或 w。

输出格式：

单独的一行 最大可能取得的珠子数。

样例输入

29

wwwbbrwrbrbrrbrbrwrwwrbwrwrrb

样例输出

11

代码：

#include<iostream>

using namespace std;

int jishu(int i, int n,char \*p)//求在i处断开收集到的珠子的个数

{

//向右收集珠子

int j = i; int k; int a=0, b=0;

while ((p[j] == 'w')&&(a<n))//当断口处为w时可以不断地收集到珠子，直至收集到的不是w的珠子或已将所有的珠子收集完

{

a++;

j=(j+1)>=n?(j+1-n):(j+1);

}

k = j;

if (a != n)//当珠子没有收集完时继续收集第一个出现的不是w类型的珠子

{

do{

a++; j = (j + 1) >= n ? (j +1- n) : (j + 1);

} while ((p[j] == 'w') || (p[j] == p[k])&&(a<n));

}

if (a==n)//如果从一个方向可以将珠子收集完那么就不用从另一个方向收集了

return a;

//向左收集珠子

j = (i - 1) >= 0 ? (i - 1) : (i+n-1);

while ((p[j] == 'w') && (b<n))

{

b++;

j=(j-1)>=0?(j-1):(j+n-1);

}

if (b != n)

{

k = j;

do{

b++; j = (j - 1) >= 0 ? (j - 1) : (j + n - 1);

} while (((p[j] == 'w') || (p[j] == p[k])) && (b < n));

}

else

b = 0;

return (a+b)>n?n:(a+b);//返回收集到的珠子的总数

}

int main()

{

int n, max;//n为输入珠子的数目

cin >> n;//输入珠子数

char \*p = new char[n];//定义动态字符数组储存珠子颜色字符串

cin.get();

cin.get(p, n+1);//输入颜色字符串

max = jishu(0, n, p);//从第一个珠子前处断开开始收珠子最多收到的个数记为max

for (int i = 1; i < n; i++)//循环找出收集最多的珠子的个数（i为开始收珠子的位置）

{

if (p[i] != 'w')//当断口处为‘w’时，收集到的珠子的个数与前一个断口处收集的个数相同，故不做循环

{

if (jishu(i, n, p)>max)

max = jishu(i, n, p);

}

}

cout << max ;

return 0;

}

F. 如果一个数从左往右读和从右往左读都是一样，那么这个数就叫做“回文数”。例如，12321就是一个回文数，而77778就不是。当然，回文数的首和尾都应是非零的，因此0220就不是回文数。

事实上，有一些数（如21），在十进制时不是回文数，但在其它进制（如二进制时为10101）时就是回文数。

编一个程序，从文件读入两个十进制数N (1 <= N <= 15)S (0 < S < 10000)然后找出前N个满足大于S且在两种或两种以上进制（二进制至十进制）上是回文数的十进制数，输出到文件上。

本问题的解决方案不需要使用大于32位的整型

输入格式

只有一行，用空格隔开的两个数N和S。

输出格式

N行, 每行一个满足上述要求的数，并按从小到大的顺序输出.

样例输入

3 25

样例输出

26

27

28

代码：

#include <iostream>

using namespace std;

const int maxn=15;

const int range=10^5;

int c[range];

int fit[maxn];

bool fun(int a,int k){

int bit=0;

while(a){

c[bit]=a%k;

a=a/k;

bit++;

}

int st=0,en=bit-1;

while(st<en)

{

if(c[st]!=c[en]) return false;

st++;en--;

}

//if(c[0]==0) return false;

return true;

}

int f(int a) {

int j=0;

for(int i=2;i<=10;i++) {

if(fun(a,i))

j++;

}

return j;

}

int main() {

int N,S;

cin>>N>>S;

int ans=0;

int i=1,l=0;

while(ans<N){

int a=S+i;

int q=f(a);

if(q>=2)

{fit[l]=a;l++;ans++;}

i++;

}

for(int m=0;m<l-1;m++){

cout<<fit[m]<<endl;

}

cout<<fit[l-1];

return 0;

}

G. 郭铮鹏认为排序是一种很频繁的计算任务，所以他考虑了一个简单的问题：现在最多只有三值的排序问题。一个实际的例子是，当我们给某项竞赛的优胜者按金银铜牌排序的时候。在这个任务中可能的值只有三种1，2和3。我们用交换的方法把他排成升序的。

郭铮鹏想让你写一个程序计算出，给定的一个1,2,3组成的数字序列，排成升序所需的最少交换次数。

输入格式

第一行：

奖牌个数N (1 <= N <= 1000)

第 2行到第N+1行:

每行一个数字，表示奖牌。共N行。（1..3）

输出格式

共一行，一个数字。表示排成升序所需的最少交换次数。

样例输入

9

2

2

1

3

3

3

2

3

1

样例输出

4

代码：

#include <iostream>

using namespace std;

const int maxn=1000;

int arr[maxn];

int main() {

int N,a=0,b=0; //a为1的个数，b为2的个数

scanf("%d",&N);

for(int i=0;i<N;i++) //输入

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

for(int i=0;i<N;i++) { //数1

if(arr[i]==1)

a++;

}

for(int i=0;i<N;i++) { //数2

if(arr[i]==2)

b++;

}

int d[3]={0,0,0},e[3]={0,0,0},f[3]={0,0,0}; //将数列分为三部分，分别存放1 2 3，每数列第i项存放这部分非i的个数

for(int i=0;i<a;i++) { //本应存1的

if(arr[i]==1)

d[0]++;

}

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

if(arr[i]==2)

d[1]++;

}

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

if(arr[i]==3)

d[2]++;

}

for(int i=a;i<a+b;i++) { //本应存2

if(arr[i]==1)

e[0]++;

}

for(int i=a;i<a+b;i++) {

if(arr[i]==2)

e[1]++;

}

for(int i=a;i<a+b;i++) {

if(arr[i]==3)

e[2]++;

}

for(int i=a+b;i<N;i++) { //本应存3

if(arr[i]==1)

f[0]++;

}

for(int i=a+b;i<N;i++) {

if(arr[i]==2)

f[1]++;

}

for(int i=a+b;i<N;i++) {

if(arr[i]==3)

f[2]++;

}

int c1=d[1]<e[0]?d[1]:e[0];

int c2=d[2]<f[0]?d[2]:f[0];

int c3=e[2]<f[1]?e[2]:f[1];

int c4=d[1]+d[2]-c1-c2;

printf("%d",c1+c2+c3+2\*c4);

return 0;

}

## （二）Training Day 2

A.郭铮鹏经营了一家冒牌可乐工厂，生意非常红火。在接下去的 N 个月里，第 i 个月需要向社会提供 Ai吨可乐。可乐的生产受到很多因素的影响，所以每个月的生产成本是变化的，其中第 i 个月的成本是每吨 Ci 元。 郭铮鹏可以提前里把可乐做好，存在仓库里，等需要的时候再拿出来卖。存储在仓库里的可乐，每吨可乐存放一个月需要支付 S 元的维护费用，存放的时间可以任意长。假设工厂的产量是无限的，存储可乐的仓库也是无限大的。请问为了满足订单的需要，郭铮鹏生产这些可乐最少要花多少钱？

Input

第一行：两个整数 N 和 S，1 ≤ N ≤ 10000, 1 ≤ S ≤ 100 第二行到第 N + 1 行：第 i + 1 行有两个整数 Ci 和 Ai，1 ≤ Ci ≤ 5000, 1 ≤ Ai ≤ 10000

Output

单个整数：表示生产可乐的最小总费用

Sample Input

4 5

88 200

89 400

97 300

91 500

Sample Output

126900

代码：

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn=10000;

int C[maxn],A[maxn],P[maxn];

int main(){

int N,S;

scanf("%d%d",&N,&S);

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

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

P[1]=C[1];

for(int i=2;i<=N;i++)

P[i]=(P[i-1]+S<C[i])?(P[i-1]+S):C[i];

long long sum=0;

for(int i=1;i<=N;i++){

sum+=P[i]\*A[i];

}

printf("%lld",sum);

return 0;

}

B.郭铮鹏开车回家，遇到了双十一节，那么就顺路买点果汁吧。回家的路程一共有 E 公里，这一路上会经过 N 家商店，第 i 家店里有 Fi 吨果汁，售价为每吨 Ci 元。郭铮鹏打算买 K 吨果汁，他知道商家的库存是足够的，至少所有店的库存总和不会少于 K。除了购买果汁要钱，运送果汁也是要花油钱的，郭铮鹏的卡车上如果装着 X 吨果汁，那么他行驶一公里会花掉 X 元，行驶 D 公里需要 DX 元。已知第 i 家店距郭铮鹏所在的起点有 Xi 公里，那么郭铮鹏在哪些商店买果汁运回家，才能做到最省钱呢？

Input

第一行：三个整数 K，E 和 N，1 ≤ K ≤ 100, 1 ≤ E ≤ 350, 1 ≤ N ≤ 100 第二行到第 N + 1 行：第 i + 1 行有三个整数 Xi，Fi 和 Ci，0 < Xi < E, 1 ≤ Fi ≤ 100, 1 ≤ Ci ≤ 10^6

Output

单个整数：表示购买和运送果汁的最小费用之和

Sample Input

2 5 3

3 1 2

4 1 2

1 1 1

Sample Output

7

代码：

#include<iostream>

#include<algorithm>

using namespace std;

struct shops{

int F;

int C;

int X;

int P;

}shop[101];

bool ascending(shops a,shops b){

return a.P<b.P;

}

int main(){

int K,E,N;

scanf("%d%d%d",&K,&E,&N);

for(int i=1;i<=N;i++){

scanf("%d%d%d",&shop[i].X,&shop[i].F,&shop[i].C);

shop[i].P=E-shop[i].X+shop[i].C;

}

sort(shop+1,shop+N,ascending);

long long sum=0;

for(int i=1;i<=N;i++){

int j=shop[i].F<K?shop[i].F:K;

sum+=j\*shop[i].P;

K-=j;

if(K<=0) break;

}

printf("%lld",sum);

return 0;

}

C.郭铮鹏开了家公司，已经连续运作了N天。他们在第i天获得了Ai元的利润，不过有些天是亏钱的，这种情况下利润就是一个负数。郭铮鹏想写个大新闻，吹嘘他的惊人业绩。请你帮助他选出一段连续的日子，长度不限，但至少要包括一天，使得公司在这段日子里的利润之和最大。

Input

第一行：单个整数N，1 ≤ N ≤ 10^5 第二行到第N + 1 行：第i + 1 行有一个整数Ai，−1000 ≤ Ai ≤ 1000

Output

单个整数：表示最大的利润之和

Sample Input

7

-3

4

9

-2

-5

8

-3

Sample Output

14

代码：

#include<stdio.h>

const int maxn=100001;

int A[maxn]; //记录利润

int D[maxn];

int main(){

int N;

long long most=0;

bool x=0;

scanf("%d",&N);

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

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

D[1]=A[1];

for(int j=2;j<=N;j++){

D[j]=(A[j]>A[j]+D[j-1])?A[j]:A[j]+D[j-1];

}

most=D[1];

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

if(D[i]>most) most=D[i];

printf("%lld",most);

return 0;

}

G.最近电视里有很多郭铮鹏想看的节目。节目一共有 N 个，第 i 个节目从时刻 Si 开始，到时刻 Ti结束。郭铮鹏有两台录像机，每台录像机可以记录时间不冲突的节目。如果一个节目的结束时间正好是另一个节目的开始时间，那么这两个节目是可以用一台录像机的。请问郭铮鹏用这两台录像机，最多可以录多少节目呢？

Input

第一行：一个整数 N，1 ≤ N ≤ 150 第二行到第 N + 1 行：第 i + 1 行有两个整数 Si 和 Ti，0 ≤ Si < Ti ≤ 10^9

Output

单个整数：表示郭铮鹏最多可以录多少节目

Sample Input

6

0 3

6 7

3 10

1 5

2 8

1 9

Sample Output

4

代码：

#include<iostream>

#include<algorithm>

using namespace std;

int p1,p2,N,cnt;

struct prog{

int st;

int en;

}pro[155];

bool compare(prog a,prog b){

return a.en<b.en;

}

int main()

{

scanf("%d",&N);

for(int i=0;i<N;i++)scanf("%d%d",&pro[i].st,&pro[i].en); //输入时间

sort(pro,pro+N,compare);

p1=-1;p2=-1;cnt=0;

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

{

if((pro[i].st<p1)&&(pro[i].st<p2))continue;

if((pro[i].st>=p1)&&(pro[i].st<p2))

{

cnt++;

p1=pro[i].en;

continue;

}

if((pro[i].st>=p2)&&(pro[i].st<p1))

{

cnt++;

p2=pro[i].en;

continue;

}

if(pro[i].st>=p1&&pro[i].st>=p2&&pro[i].st-p1<=pro[i].st-p2)

{

cnt++;

p1=pro[i].en;

continue;

}

else{

cnt++;

p2=pro[i].en;

continue;

}

}

printf("%d",cnt);

}

## （三）Training Day 3

A. 一个等差数列是一个能表示成a, a+b, a+2b,..., a+nb (n=0,1,2,3,...)的数列。

在这个问题中a是一个非负的整数，b是正整数。写一个程序来找出在双平方数集合(双平方数集合是所有能表示成p的平方 + q的平方的数的集合,其中p和q为非负整数)S中长度为n的等差数列。

第一行: N(3<= N<=25),要找的等差数列的长度。

第二行: M(1<= M<=250),搜索双平方数的上界0 <= p,q <= M。   
输出格式:

如果没有找到数列,输出“NONE”。

如果找到了，输出一行或多行, 每行由二个整数组成:a,b。 a为等差数列的第一个值，b为等差数列的公差。

这些行应该先按b排序再按a排序。

所求的等差数列将不会多于10,000个。

样例输入：

5

7

样例输出：

1 4

37 4

2 8

29 8

1 12

5 12

13 12

17 12

5 20

2 24

代码：

#include<iostream>

#include<cstring>

#include <algorithm>

using namespace std;

const int maxn=150000;

bool is[maxn];

int b[maxn];

int N,M,w=0,k=0;

int square(int M){

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

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

is[i\*i+j\*j]=1;

for(int i=0;i<=M\*M\*2;i++)

if(is[i])

b[k++]=i;

}

int main(){

scanf("%d%d",&N,&M);

square(M);

int top=(M\*M\*2-0)/(N-1);

for (int d=1;d<=top;d++){

for(int i=0;b[i]+(N-1)\*d<=M\*M\*2&&i<k;i++){

bool f=1;

for(int j=1;j<N&&f;j++)

if(!is[b[i]+j\*d]){

f=0;

}

if(f){

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

w++;

}

}

}

if(!w) printf("NONE");

}

B.gzp有三个容量分别是A,B,C升的桶，A,B,C分别是三个从1到20的整数， 最初，A和B桶都是空的，而C桶是装满牛奶的。有时，农民把牛奶从一个桶倒到 另一个桶中，直到被灌桶装满或原桶空了。当然每一次灌注都是完全的。由于节约， 牛奶不会有丢失

写一个程序去帮助gzp找出当A桶是空的时候，C桶中牛奶所剩量的所有可能性。

INPUT FORMAT:

单独的一行包括三个整数A,B和C。

OUTPUT FORMAT:

只有一行，升序地列出当A桶是空的时候，C桶牛奶所剩量的所有可能性

输入：8 9 10

输出：1 2 8 9 10

代码：

#include <iostream>

using namespace std;

const int maxn=21;

int idx[maxn][maxn];

int A,B,C;

void dfs(int m,int n){

int h=C-m-n; //B当前的量

if(idx[m][n]==1)return; //标记已到达的状态

if(m>A||h>B||n>C) return; //当容器里的量超过体积时return

idx[m][n]=1;

//printf("idx[%d][%d]=%d h=%d\n",m,n,idx[m][n],h);

if(m>C-n)dfs(m-C+n,C); //A->C

else dfs(0,m+n);

if(n>A-m)dfs(A,n-A+m); //C->A

else dfs(m+n,0);

if(m>B-h)dfs(m-B+h,n); //A->B

else dfs(0,n);

if(n>B-h)dfs(m,n-B+h); //C->B

else dfs(m,0);

if(h>A-m)dfs(A,n); //B->A

else dfs(m+h,n);

if(h>C-n)dfs(m,C); //B->C

else dfs(m,h+n);

}

int main(){

scanf("%d%d%d",&A,&B,&C);

dfs(0,C);

bool f=false;

for(int k=0;k<=C;k++){

if(idx[0][k]==1) {

if(!f){

printf("%d",k);

f=true;

}

else printf(" %d",k);

}

}

return 0;

}

D.考虑将如此安排在一个 3 x 3 行列中的九个时钟:

|-------| |-------| |-------|

| | | | | | |

|---O | |---O | | O |

| | | | | |

|-------| |-------| |-------|

A B C

|-------| |-------| |-------|

| | | | | |

| O | | O | | O |

| | | | | | | | |

|-------| |-------| |-------|

D E F

|-------| |-------| |-------|

| | | | | |

| O | | O---| | O |

| | | | | | | |

|-------| |-------| |-------|

G H I

目标要找一个最小的移动顺序将所有的指针指向12点。下面原表格列出了9种不同的旋转指针的方法，每一种方法都叫一次移动。选择1到9号移动方法，将会使在表格中对应的时钟的指针顺时针旋转90度。

移动方法 受影响的时钟

1 ABDE

2 ABC

3 BCEF

4 ADG

5 BDEFH

6 CFI

7 DEGH

8 GHI

9 EFHI

Example

9 9 12 9 12 12 9 12 12 12 12 12 12 12 12

6 6 6 5 -> 9 9 9 8-> 9 9 9 4 -> 12 9 9 9 -> 12 12 12

6 3 6 6 6 6 9 9 9 12 9 9 12 12 12

[但这可能不是正确的方法，请看下面]

INPUT FORMAT:

第1-3行: 三个空格分开的数字，每个数字表示一个时钟的初始时间，3,6,9,12。数字的含意和上面第一个例子一样。

OUTPUT FORMAT:

单独的一行包括一个用空格分开的将所有指针指向12:00的最短移动顺序的列表。

如果有多种方案，输出那种使其连接起来数字最小的方案。(举例来说5 2 4 6 < 9 3 1 1)。

样例：

In：9 9 12

6 6 6

6 3 6

Out : 4 5 8 9

#include<iostream>

#include <cstring>

using namespace std;

int a,b,c,d,e,f,g,h,i; //记录每中移动的次数

int dot,k,l,bol=0;

int clocks[10],s[10];

//定义每种移动

void first(){

clocks[1]++,clocks[2]++,clocks[4]++,clocks[5]++;

return;

}

void second(){

clocks[1]++,clocks[2]++,clocks[3]++;

return;

}

void third(){

clocks[2]++,clocks[3]++,clocks[5]++,clocks[6]++;

return;

}

void fourth(){

clocks[1]++,clocks[4]++,clocks[7]++;

return;

}

void fifth(){

clocks[2]++,clocks[4]++,clocks[5]++,clocks[6]++,clocks[8]++;

return;

}

void sixth(){

clocks[3]++,clocks[6]++,clocks[9]++;

return;

}

void seventh(){

clocks[4]++,clocks[5]++,clocks[7]++,clocks[8]++;

return;

}

void eighth(){

clocks[7]++,clocks[8]++,clocks[9]++;

return;

}

void ninth(){

clocks[5]++,clocks[6]++,clocks[8]++,clocks[9]++;

return;

}

int main(){

for(int z=1;z<10;z++){ //输入

scanf("%d",&dot);

clocks[z]=dot/3;

}

for(a=1;a<5;a++){

first();

for(b=1;b<5;b++){

second();

for(c=1;c<5;c++){

third();

for(d=1;d<5;d++){

fourth();

for(e=1;e<5;e++){

fifth();

for(f=1;f<5;f++){

sixth();

for(g=1;g<5;g++){

seventh();

for(h=1;h<5;h++){

eighth();

for(i=1;i<5;i++){

ninth();

for(k=1;k<=9;k++){ //九种操作运行完后 是否全部变为12 否 进入下一次循环，然后返回上一层循环，依次

if(clocks[k]%4!=0) break;

}

if(k==10) //全部变为12

{

memset(s,0,sizeof(s));

if(a!=4)s[1]=a;

if(b!=4)s[2]=b;

if(c!=4)s[3]=c;

if(d!=4)s[4]=d;

if(e!=4)s[5]=e;

if(f!=4)s[6]=f;

if(g!=4)s[7]=g;

if(h!=4)s[8]=h;

if(i!=4)s[9]=i;

for(l=1;l<=9;l++)

{

for(k=1;k<=s[l]%4;k++)

{

if(bol)cout<<" ";

bol=1;

cout<<l;

}

}

return 0;

}

}

}

}

}

}

}

}

}

}

return 0;

}

F.GZP以拥有世界上最健康的猪为傲。他知道每种饮食中所包含的猪所需的最低的维他命量是多少。请你帮助GZP喂养他的猪，以保持它们的健康，使喂给猪的饲料的种数最少。

给出猪所需的最低的维他命量，输出喂给猪需要哪些种类的饲料，且所需的饲料剂量最少。

维他命量以整数表示，每种饲料最多只能对猪使用一次，数据保证存在解。

INPUT FORMAT:

第1行：一个整数V(1<=V<=25)，表示需要的维他命的种类数。

第2行：V个整数(1<=每个数<=1000)，表示猪每天需要的每种维他命的最小量。

第3行：一个整数G(1<=G<=15)，表示可用来喂猪的饲料的种数。

下面G行，第n行表示编号为n饲料包含的各种维他命的量的多少。

OUTPUT FORMAT:

输出文件只有一行，包括

猪必需的最小的饲料种数P

后面有P个数，表示所选择的饲料编号（按从小到大排列）。

如果有多个解，输出饲料序号最小的（即字典序最小）。

SAMPLE INPUT

4

100 200 300 400

3

50 50 50 50

200 300 200 300

900 150 389 399

SAMPLE OUTPUT

2 1 3

代码：

#include <iostream>

using namespace std;

int a[20],visit[20]; //a 记录使用的饲料序列，visit标记是否使用

int v[30],food[20][30],v2[30];

int bool1,bool2,t;

int V,G;

void DFS(int x,int k)

{

int j;

if(x==k)

{

bool1=1;

for(t=1;t<=V;t++)

if(v[t]>v2[t])

{

bool1=0;

break;

}

if(bool1==1)

{

cout<<k;

for(t=1;t<=k;t++)

{

cout<<" "<<a[t];

}

bool2=1;

}

return ;

}

x++;

for(j=a[x-1]+1;j<=G;j++)

{

if(visit[j]==0)

{

visit[j]=1;

a[x]=j;

for(t=1;t<=V;t++)

{

v2[t]+=food[j][t];

}

DFS(x,k);

if(bool2==1)

break;

visit[j]=0;

for(t=1;t<=V;t++)

{

v2[t]-=food[j][t];

}

}

}

}

int main()

{

int i,j,k;

//输入

scanf("%d",&V);

for(i=1;i<=V;i++)

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

scanf("%d",&G);

for(i=1;i<=G;i++)

for(j=1;j<=V;j++)

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

bool2=0;

for(k=1;k<=G;k++) //k表示使用饲料的总数

{

for(i=1;i<=G;i++) //i表示从第一种饲料开始遍历

{

if(visit[i]==0)

{

visit[i]=1;

a[1]=i;

for(t=1;t<=V;t++)

v2[t]+=food[i][t]; //使用第i种饲料后，共组成的元素含量

DFS(1,k);

if(bool2==1)

break;

visit[i]=0;

for(t=1;t<=V;t++)

{

v2[t]-=food[i][t];

}

}

}

if(bool2==1)

break;

}

return 0;

}

## （四）Training Day 4

A.两个机器人逃跑到了森林里。大神gzp开始用他的专家技术追捕这两个机器人。你的任务是模拟他们的行为(机器人和gzp)。

追击在10x10的平面网格内进行。一个格子可以是：

一个障碍物, 两个机器人(它们总在一起), 或者gzp.

两个机器人和gzp可以在同一个格子内(当他们相遇时)，但是他们都不能进入有障碍的格子。

一个格子可以是：

. 空地

\* 障碍物

C 两个机器人

F gzp

这里有一个地图的例子：

\*...\*.....

......\*...

...\*...\*..

..........

...\*.F....

\*.....\*...

...\*......

..C......\*

...\*.\*....

.\*.\*......

机器人在地图里以固定的方式游荡。每分钟，它们可以向前移动或是转弯。如果前方无障碍(地图边沿也是障碍)，它们会按照原来的方向前进一步。否则它们会用这一分钟顺时针转90度。 同时，它们不会离开地图。

gzp深知机器人的移动方法，他也这么移动。

每次(每分钟)gzp和两个机器人的移动是同时的。如果他们在移动的时候穿过对方，但是没有在同一格相遇，我们不认为他们相遇了。当他们在某分钟末在某格子相遇，那么追捕结束。

读入十行表示大神Gzp,两个机器人和所有障碍的位置的地图。每行都个包含10个字符，表示的含义和上面所说的相同，你可以确定地图中个有一个'F'和一个'C'.'F'和'C'一开始不会处于同一个格子中。

计算大神Gzp需要多少分钟来抓住他的机器人，假设机器人和大神Gzp一开始的行动方向都是正北（即上）。 如果Gzp和机器人永远不会相遇，输出0。

INPUT FORMAT:

第1-10行:

每行10个字符，表示如上文描述的地图。

OUTPUT FORMAT:

输出一个数字，表示Gzp需要多少时间才能抓住机器人们。如果Gzp无法抓住机器人，则输出0。

SAMPLE INPUT

\*...\*.....

......\*...

...\*...\*..

..........

...\*.F....

\*.....\*...

...\*......

..C......\*

...\*.\*....

.\*.\*......

SAMPLE OUTPUT

49

代码：

#include <iostream>

using namespace std;

const int size=12; //定12是为了防止越界

bool map[size][size]={false}; //12\*12 为FALSE 即在10\*10之外的也会false

int gx,gy,rx,ry,min; //坐标，g Gzp r：机器人 min是记录分钟数的

char dirg='u',dirr='u'; //表示 GZP和机器人的朝向

void drawmap(){ //绘制地图

for(int i=1;i<11;i++)

for(int j=1;j<11;j++){

char t;

scanf("%c",&t); //记录GZP所在

if(t=='F') {

gx=i;

gy=j;

}

if(t=='C'){ //记录机器人所在

rx=i;

ry=j;

}

if (t!='\*') map[i][j]=true;

}

}

void GzpMove(){ //GZP的每分钟可能出现的移动情况

if(dirg=='u'&&map[gx-1][gy]==false){ //向上走不了向右 但不走 下同

dirg='r';

return;

}

if(dirg=='r'&&map[gx][gy+1]==false){ //向右走不了向下

dirg='d';

return;

}

if(dirg=='d'&&map[gx+1][gy]==false){ //向下走不了向左

dirg='l';

return;

}

if(dirg=='l'&&map[gx][gy-1]==false){ //向右走不了向下

dirg='u';

return;

}

if(dirg=='u'&&map[gx-1][gy]==true){ //向上走

dirg='u';

gx-=1;

return;

}

if(dirg=='r'&&map[gx][gy+1]==true){ //向you走

dirg='r';

gy+=1;

return;

}

if(dirg=='d'&&map[gx+1][gy]==true){ //向xia走

dirg='d';

gx+=1;

return;

}

if(dirg=='l'&&map[gx][gy-1]==true){ //向zuo走

dirg='l';

gy-=1;

return;

}

}

void RobortMove(){ //机器人的每分钟可能出现的移动情况

if(dirr=='u'&&map[rx-1][ry]==false){ //向上走不了向右

dirr='r';

return;

}

if(dirr=='r'&&map[rx][ry+1]==false){ //向右走不了向下

dirr='d';

return;

}

if(dirr=='d'&&map[rx+1][ry]==false){ //向下走不了向左

dirr='l';

return;

}

if(dirr=='l'&&map[rx][ry-1]==false){ //向右走不了向下

dirr='u';

return;

}

if(dirr=='u'&&map[rx-1][ry]==true){ //向上走

dirr='u';

rx-=1;

return;

}

if(dirr=='r'&&map[rx][ry+1]==true){ //向you走

dirr='r';

ry+=1;

return;

}

if(dirr=='d'&&map[rx+1][ry]==true){ //向xia走

dirr='d';

rx+=1;

return;

}

if(dirr=='l'&&map[rx][ry-1]==true){ //向zuo走

dirr='l';

ry-=1;

return;

}

}

int main(){

drawmap();

for (int min=1;min<=1000;min++){ //这个我也不知道为什么用1000控制

GzpMove();

RobortMove();

if((gx==rx)&&(gy==ry)){ //是否相遇

printf("%d",min);

return 0; //这个是不是应该放在后面

}

if (min==1000) printf("%d",0);

}

}

D.Gzp每年有很多栅栏要修理。他总是骑着马穿过每一个栅栏并修复它破损的地方。

Gzp是一个与其他大神一样懒的人。他讨厌骑马，因此从来不两次经过一个栅栏。你必须编一个程序，读入栅栏网络的描述，并计算出一条修栅栏的路径，使每个栅栏都恰好被经过一次。Gzp能从任何一个顶点(即两个栅栏的交点)开始骑马，在任意一个顶点结束。

每一个栅栏连接两个顶点，顶点用1到500标号(虽然有的小院并没有500个顶点)。一个顶点上可连接任意多(>=1)个栅栏。两顶点间可能有多个栅栏。所有栅栏都是连通的(也就是你可以从任意一个栅栏到达另外的所有栅栏)。

你的程序必须输出骑马的路径(用路上依次经过的顶点号码表示)。我们如果把输出的路径看成是一个500进制的数，那么当存在多组解的情况下，输出500进制表示法中最小的一个 (也就是输出第一位较小的，如果还有多组解，输出第二位较小的，等等)。

输入数据保证至少有一个解。

INPUT FORMAT：

第1行: 一个整数F(1 <= F <= 1024)，表示栅栏的数目

第2到F+1行: 每行两个整数i, j(1 <= i,j <= 500)表示这条栅栏连接i与j号顶点。

OUTPUT FORMAT：

输出应当有F+1行，每行一个整数，依次表示路径经过的顶点号。注意数据可能有多组解，但是只有上面题目要求的那一组解是认为正确的。

SAMPLE INPUT

9

1 2

2 3

3 4

4 2

4 5

2 5

5 6

5 7

4 6

SAMPLE OUTPUT

1

2

3

4

2

5

4

6

5

7

代码：

#include <iostream>

#include <stack>

using namespace std;

const int MAX = 505;

int fence[MAX][MAX];

int degree[MAX];

stack<int> trace;

int n;

int max(int a, int b){

return a>b? a: b;

}

void Eular(int k){

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

if(fence[k][i]){

fence[k][i]--;

fence[i][k]--;

Eular(i);

}

trace.push(k);

}

int main(){

int i,c,a,b;

cin>>c;

n = 0;

for(i=0; i<c; ++i){

cin>>a>>b;

fence[a][b]++;

fence[b][a]++;

degree[a]++;

degree[b]++;

n = max(n, a);

n = max(n ,b);

}

bool find = false;

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

if(degree[i]%2 == 1){

find = true;

Eular(i);

break;

}

if(!find)

Eular(1);

//output

while(!trace.empty()){

cout<<trace.top()<<endl;

trace.pop();

}

return 0;

}

E. gzp被选为他们社团的主席！他其中一个竞选承诺就是在社团内建立起互联网，并连接到所有的机房。当然，他需要你的帮助。

gzp已经给他的机房安排了一条高速的网络线路，他想把这条线路共享给其他机房。为了使花费最少，他想铺设最短的光纤去连接所有的机房。你将得到一份各机房之间连接费用的列表，你必须找出能连接所有机房并所用光纤最短的方案。每两个机房间的距离不会超过100000。

INPUT FORMAT:

第一行： 机房的个数，N（3<=N<=100）。

第二行..结尾: 接下来的行包含了一个N\*N的矩阵,表示每个机房之间的距离。理论上，他们是N行，每行由N个用空格分隔的数组成，实际上，他们每行限制在80个字符以内，因此，某些行会紧接着另一些行。当然，对角线将会是0，因为线路从第i个机房到它本身的距离在本题中没有意义。

OUTPUT FORMAT:

只有一个输出，是连接到每个机房的光纤的最小长度和。

SAMPLE INPUT

4

0 4 9 21

4 0 8 17

9 8 0 16

21 17 16 0

SAMPLE OUTPUT

28

代码：

#include<iostream>

const int maxn=110;

const int size=10010; //即矩阵的最大规模

struct bond{

int start;

int end;

int dist;

}ho[size]; //表示两个机房之间的关系，关系不超过矩阵规模的一般

int cp[maxn]={0};

int N,num=0;

bool isfull(){

for (int z=1;z<=N;z++){

if(cp[z]==0) {

return false;

}

}

return true;

}

int main(){

scanf("%d",&N);

for(int s=1;s<=N;s++)

for(int e=1;e<=N;e++){

int dis;

scanf("%d",&dis);

if(s<e){ //确定机房间的关系

num++;

ho[num].start=s;

ho[num].end=e;

ho[num].dist=dis;

}

}

long long shortest=0;

int al;

cp[1]=1; //prim 算法

while(!isfull()){ //当机房未连接完毕时

int least=100000;

for(int x=1;x<=N;x++){

if(cp[x]==1){ //找一个被联结了的机房

for(int y=1;y<=N;y++){

if(cp[y]==0){ //找一个还没有被联结的机房

for(int w=1;w<=num;w++){ //通过上下这几个循环找当前状态下可连接的最短边

if((ho[w].start==x&&ho[w].end==y)||(ho[w].start==x&&ho[w].end==y)){

if(ho[w].dist<least){

least=ho[w].dist; //符合条件更新最小值

al=y; //记下当前最小边长的另一个端点，即另一个机房

}

}

}

}

}

}

}

cp[al]=1; //新机房连接完毕

shortest+=least;

}

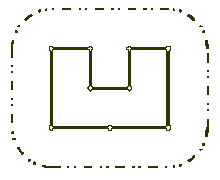
printf("%lld",shortest);

return 0;

}

## （五）Training Day 5

B.Once upon a time there was a greedy King who ordered his chief Architect to build a wall around the King's castle. The King was so greedy, that he would not listen to his Architect's proposals to build a beautiful brick wall with a perfect shape and nice tall towers. Instead, he ordered to build the wall around the whole castle using the least amount of stone and labor, but demanded that the wall should not come closer to the castle than a certain distance. If the King finds that the Architect has used more resources to build the wall than it was absolutely necessary to satisfy those requirements, then the Architect will loose his head. Moreover, he demanded Architect to introduce at once a plan of the wall listing the exact amount of resources that are needed to build the wall.  
Your task is to help poor Architect to save his head, by writing a program that will find the minimum possible length of the wall that he could build around the castle to satisfy King's requirements.

[](http://acm.hdu.edu.cn/data/images/1348-1.gif)

The task is somewhat simplified by the fact, that the King's castle has a polygonal shape and is situated on a flat ground. The Architect has already established a Cartesian coordinate system and has precisely measured the coordinates of all castle's vertices in feet.

Input

The first line of the input file contains two integer numbers N and L separated by a space. N (3 <= N <= 1000) is the number of vertices in the King's castle, and L (1 <= L <= 1000) is the minimal number of feet that King allows for the wall to come close to the castle.  
  
Next N lines describe coordinates of castle's vertices in a clockwise order. Each line contains two integer numbers Xi and Yi separated by a space (-10000 <= Xi, Yi <= 10000) that represent the coordinates of ith vertex. All vertices are different and the sides of the castle do not intersect anywhere except for vertices.

Output

Write to the output file the single number that represents the minimal possible length of the wall in feet that could be built around the castle to satisfy King's requirements. You must present the integer number of feet to the King, because the floating numbers are not invented yet. However, you must round the result in such a way, that it is accurate to 8 inches (1 foot is equal to 12 inches), since the King will not tolerate larger error in the estimates.  
  
This problem contains multiple test cases!  
  
The first line of a multiple input is an integer N, then a blank line followed by N input blocks. Each input block is in the format indicated in the problem description. There is a blank line between input blocks.  
  
The output format consists of N output blocks. There is a blank line between output blocks.

Sample Input

1

9 100

200 400

300 400

300 300

400 300

400 400

500 400

500 200

350 200

200 200

Sample Output

1628

**代码：**

#include <iostream>

#include <string.h>

#include <algorithm>

#include <math.h>

using namespace std;

const int M=1000;

const float pi=3.1415926;

int top;

struct node {

double x,y;

}p[M],stack[M];

//计算两点间距离。

double L(node a,node b)

{

double dx=a.x-b.x,dy=a.y-b.y;

return hypot(dx,dy);

}

//计算叉积的大小。

double multi(node a,node b,node c)

{

return (a.x-c.x)\*(b.y-c.y)-(b.x-c.x)\*(a.y-c.y);

}

//计算左转还是右转，大于零就是左转，小于零就是右转。

int cmp(node a,node b)

{

if(multi(a,b,p[0])>0)

return 1;

if(multi(a,b,p[0])==0&&L(a,p[0])<L(b,p[0]))

return 1;

return 0;

}

void GS(node p[],node stack[],int n)

{

int i,k=0;

node temp;

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

{

if(p[i].y<p[k].y||((p[i].y==p[k].y)&&(p[i].x<p[k].x)))

k=i;

}

temp=p[0];

p[0]=p[k];

p[k]=temp;

sort(p+1,p+n,cmp);

int top=2;

stack[0]=p[0],stack[1]=p[1],stack[2]=p[2];

for(i=3;i<n;i++)

{

while(top>1&&multi(p[i],stack[top],stack[top-1])>=0)

top--; //如果右转，栈顶元素出栈。

stack[++top]=p[i]; //找到左转，元素入栈。

}

}

int main()

{

double t;

int n,i,m; //n是个数 m是半径

while(scanf("%d%d",&n,&m)!=EOF)

{ t=0;

memset(p,0,sizeof(p));

memset(stack,0,sizeof(stack));

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

{

scanf("%lf %lf",&p[i].x,&p[i].y);

}

GS(p,stack,n);

stack[top+1]=stack[0];

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

{

t+=L(stack[i],stack[i+1]);

}

t=t+2\*m\*pi;

printf("%.0lf\n",t);

}

return 0;

}

## （六）Training Day 6

A. A numeric sequence of *ai* is ordered if *a1* < *a2* < ... < *aN*. Let the subsequence of the given numeric sequence (*a1*, *a2*, ..., *aN*) be any sequence (*ai1*, *ai2*, ..., *aiK*), where 1 <= *i1* < *i2* < ... < *iK* <= *N*. For example, sequence (1, 7, 3, 5, 9, 4, 8) has ordered subsequences, e. g., (1, 7), (3, 4, 8) and many others. All longest ordered subsequences are of length 4, e. g., (1, 3, 5, 8).  
  
Your program, when given the numeric sequence, must find the length of its longest ordered subsequence.

Input

The first line of input file contains the length of sequence N. The second line contains the elements of sequence - N integers in the range from 0 to 10000 each, separated by spaces. 1 <= N <= 1000

Output

Output file must contain a single integer - the length of the longest ordered subsequence of the given sequence.

Sample Input

7

1 7 3 5 9 4 8

Sample Output

4

**代码：**

#include<iostream>

#include<cstdio>

using namespace std;

const int maxn=1005;

int f[maxn]; //以第i个数结尾的上升序列包含最多的数

int num[maxn]; //存放数列（从i=1开始）

int main(){

int N;

scanf("%d",&N);

for (int k=1;k<=N;k++){

scanf("%d",&num[k]);

}

f[1]=1;

int res=1;

for(int i=2;i<=N;i++){

f[i]=1;

for(int j=1;j<i;j++){

if(num[j]<num[i]){

f[i]=max(f[j]+1,f[i]);

}

}

res=max(res,f[i]);

}

printf("%d",res);

return 0;

}

B.We give the following inductive definition of a “regular brackets” sequence:

* the empty sequence is a regular brackets sequence,
* if *s* is a regular brackets sequence, then (*s*) and [*s*] are regular brackets sequences, and
* if *a* and *b* are regular brackets sequences, then *ab* is a regular brackets sequence.
* no other sequence is a regular brackets sequence

For instance, all of the following character sequences are regular brackets sequences:

(), [], (()), ()[], ()[()]

while the following character sequences are not:

(, ], )(, ([)], ([(]

Given a brackets sequence of characters *a*1*a*2 … *an*, your goal is to find the length of the longest regular brackets sequence that is a subsequence of *s*. That is, you wish to find the largest *m* such that for indices *i*1, *i*2, …, *im* where 1 ≤ *i*1 < *i*2 < … < *im* ≤ *n*, *ai*1*ai*2 …*aim* is a regular brackets sequence.

Given the initial sequence ([([]])], the longest regular brackets subsequence is [([])].

Input

The input test file will contain multiple test cases. Each input test case consists of a single line containing only the characters (,), [, and ]; each input test will have length between 1 and 100, inclusive. The end-of-file is marked by a line containing the word “end” and should not be processed.

Output

For each input case, the program should print the length of the longest possible regular brackets subsequence on a single line.

Sample Input

((()))

()()()

([]])

)[)(

([][][)

end

Sample Output

6

6

4

0

6

**代码：**

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cmath>

using namespace std;

const int maxn=105;

char bracket[maxn];

int dp[maxn][maxn];

int ans[maxn];

int search(int l,char a[]){

for(int i=l-1;i>0;i--){

for (int j=i+1;j<=l;j++){

dp[i][j]=dp[i+1][j];

if(bracket[i]=='('||bracket[i]=='['){

for(int k=i+1;k<=j;k++){

if((bracket[i]=='('&&bracket[k]==')')||(bracket[i]=='['&&bracket[k]==']')){

dp[i][j]=max(dp[i][j],dp[i+1][k-1]+dp[k][j]+2);

}

}

}

}

}

return dp[1][l];

}

int main(){

int t=0;

while(scanf("%s",bracket+1)){

if(strcmp(bracket+1,"end")==0) break;

memset(dp,0,sizeof(dp));

int len=strlen(bracket+1);

ans[t]=search(len,bracket+1);

t++;

}

for(int z=0;z<t-1;z++){

printf("%d\n",ans[z]);

}

printf("%d",ans[t-1]);

return 0;

}

C. Gigel has a strange "balance" and he wants to poise it. Actually, the device is different from any other ordinary balance.   
It orders two arms of negligible weight and each arm's length is 15. Some hooks are attached to these arms and Gigel wants to hang up some weights from his collection of G weights (1 <= G <= 20) knowing that these weights have distinct values in the range 1..25. Gigel may droop any weight of any hook but he is forced to use all the weights.   
Finally, Gigel managed to balance the device using the experience he gained at the National Olympiad in Informatics. Now he would like to know in how many ways the device can be balanced.   
  
Knowing the repartition of the hooks and the set of the weights write a program that calculates the number of possibilities to balance the device.   
It is guaranteed that will exist at least one solution for each test case at the evaluation.

Input

The input has the following structure:   
• the first line contains the number C (2 <= C <= 20) and the number G (2 <= G <= 20);   
• the next line contains C integer numbers (these numbers are also distinct and sorted in ascending order) in the range -15..15 representing the repartition of the hooks; each number represents the position relative to the center of the balance on the X axis (when no weights are attached the device is balanced and lined up to the X axis; the absolute value of the distances represents the distance between the hook and the balance center and the sign of the numbers determines the arm of the balance to which the hook is attached: '-' for the left arm and '+' for the right arm);   
• on the next line there are G natural, distinct and sorted in ascending order numbers in the range 1..25 representing the weights' values.

Output

The output contains the number M representing the number of possibilities to poise the balance.

Sample Input

2 4

-2 3

3 4 5 8

Sample Output

2

**代码：**

#include <iostream>

#include <cstdio>

#include<cstring>

using namespace std;

const int num=21;

const int maxn=15005;

int r[num]; //钩距支点的距离

int w[num]; //砝码的重量

int dp[num][maxn];//达到挂了i个砝码之后平衡度为j（原始+7500）的状态的方案数

int main(){

int C,G;

scanf("%d%d",&C,&G);

for(int i=1;i<=C;i++){

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

}

for(int i=1;i<=G;i++){

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

}

memset(dp,0,sizeof(dp));

dp[0][7500]=1; //一个都不挂可达到平衡状态

for(int i=1;i<=G;i++){

for(int j=0;j<=15000;j++){

if(dp[i-1][j]){

for(int k=1;k<=C;k++){

dp[i][j+w[i]\*r[k]]+=dp[i-1][j];

}

}

}

}

printf("%d",dp[G][7500]);

}

D. Bessie has gone to the mall's jewelry store and spies a charm bracelet. Of course, she'd like to fill it with the best charms possible from the *N* (1 ≤ *N* ≤ 3,402) available charms. Each charm *i* in the supplied list has a weight *Wi* (1 ≤ *Wi* ≤ 400), a 'desirability' factor*Di* (1 ≤ *Di* ≤ 100), and can be used at most once. Bessie can only support a charm bracelet whose weight is no more than *M* (1 ≤ *M*≤ 12,880).

Given that weight limit as a constraint and a list of the charms with their weights and desirability rating, deduce the maximum possible sum of ratings.

Input

\* Line 1: Two space-separated integers: *N* and *M*  
\* Lines 2..*N*+1: Line *i*+1 describes charm *i* with two space-separated integers: *Wi* and *Di*

Output

\* Line 1: A single integer that is the greatest sum of charm desirabilities that can be achieved given the weight constraints

Sample Input

4 6

1 4

2 6

3 12

2 7

Sample Output

23

**代码：**

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

const int num=3405;

const int pa=12885;

int dp[pa];

int wei[num];

int val[num];

int N,M;

int main(){

scanf("%d %d",&N,&M);

for(int i=1;i<=N;i++){

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

}

memset(dp,0,sizeof(dp));

for(int i=1;i<=N;i++){

for(int j=M;j>=wei[i];j--){

dp[j]=max(dp[j],dp[j-wei[i]]+val[i]);

}

}

printf("%d",dp[M]);

}

E. Before ACM can do anything, a budget must be prepared and the necessary financial support obtained. The main income for this action comes from Irreversibly Bound Money (IBM). The idea behind is simple. Whenever some ACM member has any small money, he takes all the coins and throws them into a piggy-bank. You know that this process is irreversible, the coins cannot be removed without breaking the pig. After a sufficiently long time, there should be enough cash in the piggy-bank to pay everything that needs to be paid.   
  
But there is a big problem with piggy-banks. It is not possible to determine how much money is inside. So we might break the pig into pieces only to find out that there is not enough money. Clearly, we want to avoid this unpleasant situation. The only possibility is to weigh the piggy-bank and try to guess how many coins are inside. Assume that we are able to determine the weight of the pig exactly and that we know the weights of all coins of a given currency. Then there is some minimum amount of money in the piggy-bank that we can guarantee. Your task is to find out this worst case and determine the minimum amount of cash inside the piggy-bank. We need your help. No more prematurely broken pigs!

Input

The input consists of T test cases. The number of them (T) is given on the first line of the input file. Each test case begins with a line containing two integers E and F. They indicate the weight of an empty pig and of the pig filled with coins. Both weights are given in grams. No pig will weigh more than 10 kg, that means 1 <= E <= F <= 10000. On the second line of each test case, there is an integer number N (1 <= N <= 500) that gives the number of various coins used in the given currency. Following this are exactly N lines, each specifying one coin type. These lines contain two integers each, Pand W (1 <= P <= 50000, 1 <= W <=10000). P is the value of the coin in monetary units, W is it's weight in grams.

Output

Print exactly one line of output for each test case. The line must contain the sentence "The minimum amount of money in the piggy-bank is X." where X is the minimum amount of money that can be achieved using coins with the given total weight. If the weight cannot be reached exactly, print a line "This is impossible.".

Sample Input

3

10 110

2

1 1

30 50

10 110

2

1 1

50 30

1 6

2

10 3

20 4

Sample Output

The minimum amount of money in the piggy-bank is 60.

The minimum amount of money in the piggy-bank is 100.

This is impossible.

**代码：**

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

const int maxn=10005;

const int kind=505;

const int INF=0x3f3f3f3f;

int dp[maxn];

int P[kind],W[kind]; // p是价值 w是重量

int T;

int res[maxn]; //用于记录每种情况对应下的最大值，以便输出

int main(){

scanf("%d",&T); //几种情况

for(int i=1;i<=T;i++){

int N,E,F;

scanf("%d%d%d",&E,&F,&N); //E是纯猪重量，F是放了钱之后的猪重量 N是几种硬币

int net=F-E; //钱重量

memset(P,0,sizeof(P)); // 在有多种情况下共用一个数组记录数组，每次都要清零的！！！！！！

memset(W,0,sizeof(W));

for(int j=1;j<=N;j++){

scanf("%d%d",&P[j],&W[j]);

}

memset(dp,0x3f,sizeof(dp));

dp[0]=0;

for(int k=1;k<=N;k++){

for(int v=W[k];v<=net;v++){

dp[v]=((dp[v])>=(dp[v-W[k]]+P[k]))?(dp[v-W[k]]+P[k]):dp[v];

}

}

res[i]=dp[net];

}

for(int z=1;z<T;z++){

if(res[z]==INF)printf("This is impossible.\n");

else printf("The minimum amount of money in the piggy-bank is %d.\n",res[z]);

}

if(res[T])printf("This is impossible.");

else printf("The minimum amount of money in the piggy-bank is %d.",res[T]);

return 0;

}

## （七）Training Day 7

A. There is a famous railway station in PopPush City. Country there is incredibly hilly. The station was built in last century. Unfortunately, funds were extremely limited that time. It was possible to establish only a surface track. Moreover, it turned out that the station could be only a dead-end one (see picture) and due to lack of available space it could have only one track.   
The local tradition is that every train arriving from the direction A continues in the direction B with coaches reorganized in some way. Assume that the train arriving from the direction A has N <= 1000 coaches numbered in increasing order 1, 2, ..., N. The chief for train reorganizations must know whether it is possible to marshal coaches continuing in the direction B so that their order will be a1, a2, ..., aN. Help him and write a program that decides whether it is possible to get the required order of coaches. You can assume that single coaches can be disconnected from the train before they enter the station and that they can move themselves until they are on the track in the direction B. You can also suppose that at any time there can be located as many coaches as necessary in the station. But once a coach has entered the station it cannot return to the track in the direction A and also once it has left the station in the direction B it cannot return back to the station.

Input

The input consists of blocks of lines. Each block except the last describes one train and possibly more requirements for its reorganization. In the first line of the block there is the integer N described above. In each of the next lines of the block there is a permutation of 1, 2, ..., N. The last line of the block contains just 0.   
The last block consists of just one line containing 0.

Output

The output contains the lines corresponding to the lines with permutations in the input. A line of the output contains Yes if it is possible to marshal the coaches in the order required on the corresponding line of the input. Otherwise it contains No. In addition, there is one empty line after the lines corresponding to one block of the input. There is no line in the output corresponding to the last ``null'' block of the input.

Sample Input

5

1 2 3 4 5

5 4 1 2 3

0

6

6 5 4 3 2 1

0

0

Sample Output

Yes

No

Yes

**代码：**

#include<iostream>

#include<stack>

#include<cstdio>

using namespace std;

const int maxn=1005;

int N;

stack<int>c;

int b[maxn];

bool judge(int t,int b[]){

int i,j;

for(i=1,j=1;j<=t;i++){

if(b[j]==i)

j++;

else if(!c.empty()&&c.top()==b[j]){

c.pop();

j++;

i--;

}

else if(i<=t)

c.push(i);

else{

while(!c.empty()) c.pop();

return 0;

}

}

while(!c.empty()) c.pop();

return 1;

}

int main(){

while(scanf("%d",&N)){

if(!N) break;

while(scanf("%d",&b[1])){ //用于输入一个N下不同的出队序列

if(!b[1]) {

break;

}

for(int i=2;i<=N;i++){

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

}

if(judge(N,b)) printf("Yes\n");

else printf("No\n");

}

printf("\n");

}

return 0;

}

B.Some of Farmer John's *N* cows (1 ≤ *N* ≤ 80,000) are having a bad hair day! Since each cow is self-conscious about her messy hairstyle, FJ wants to count the number of other cows that can see the top of other cows' heads.

Each cow *i* has a specified height *hi* (1 ≤ *hi*≤ 1,000,000,000) and is standing in a line of cows all facing east (to the right in our diagrams). Therefore, cow *i* can see the tops of the heads of cows in front of her (namely cows *i*+1, *i*+2, and so on), for as long as these cows are strictly shorter than cow *i*.

Consider this example:

        =  
=       =  
=   -   =         Cows facing right -->  
=   =   =  
= - = = =  
= = = = = =  
1 2 3 4 5 6

Cow#1 can see the hairstyle of cows #2, 3, 4  
Cow#2 can see no cow's hairstyle  
Cow#3 can see the hairstyle of cow #4  
Cow#4 can see no cow's hairstyle  
Cow#5 can see the hairstyle of cow 6  
Cow#6 can see no cows at all!

Let *ci* denote the number of cows whose hairstyle is visible from cow *i*; please compute the sum of *c*1 through *cN*.For this example, the desired is answer 3 + 0 + 1 + 0 + 1 + 0 = 5.

Input

Line 1: The number of cows, *N*.   
Lines 2..N+1: Line *i*+1 contains a single integer that is the height of cow *i*.

Output

Line 1: A single integer that is the sum of *c*1 through *cN*.

Sample Input

6

10

3

7

4

12

2

Sample Output

5

**代码：**

#include<cstdio>

#include<stack>

using namespace std;

const int maxn=800005;

int h[maxn];

stack<int>cow;

int main(){

int N;

long long sum=0;

scanf("%d",&N);

scanf("%d",&h[1]);

cow.push(h[1]);

for (int i=2;i<=N;i++){

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

while(!cow.empty()&&h[i]>=cow.top()){

cow.pop();

}

sum+=cow.size();

cow.push(h[i]);

}

printf("%lld",sum);

return 0;

}

C.To avoid unsightly burns while tanning, each of the *C* (1 ≤ *C* ≤ 2500) cows must cover her hide with sunscreen when they're at the beach. Cow *i* has a minimum and maximum *SPF* rating (1 ≤ *minSPFi* ≤ 1,000; *minSPFi* ≤ *maxSPFi* ≤ 1,000) that will work. If the*SPF* rating is too low, the cow suffers sunburn; if the *SPF* rating is too high, the cow doesn't tan at all........

The cows have a picnic basket with *L* (1 ≤ *L* ≤ 2500) bottles of sunscreen lotion, each bottle *i* with an *SPF* rating *SPFi* (1 ≤ *SPFi* ≤ 1,000). Lotion bottle *i* can cover *coveri* cows with lotion. A cow may lotion from only one bottle.

What is the maximum number of cows that can protect themselves while tanning given the available lotions?

Input

\* Line 1: Two space-separated integers: *C* and *L*  
\* Lines 2..*C*+1: Line *i* describes cow *i*'s lotion requires with two integers: *minSPFi* and *maxSPFi*   
\* Lines *C*+2..*C*+*L*+1: Line *i*+*C*+1 describes a sunscreen lotion bottle *i* with space-separated integers: *SPFi* and *coveri*

Output

A single line with an integer that is the maximum number of cows that can be protected while tanning

Sample Input

3 2

3 10

2 5

1 5

6 2

4 1

Sample Output

2

**代码：**

#include<iostream>

#include<cstdio>

#include<queue>

#include<functional>

#include<vector>

#include<string>

#include<algorithm>

using namespace std;

const int maxn=2505;

int res=0;

priority\_queue<int,vector<int>,greater<int> >que;

struct cows{

int minspf;

int maxspf;

}cow[maxn];

struct lotion{

int spf;

int amt;

}lot[maxn];

bool cmp(cows a,cows b){

// return a.maxspf==b.maxspf?a.minspf<b.minspf:a.maxspf<b.maxspf; //先按最大值排，最大值一样按最小值排

return a.minspf<b.minspf;

}

bool cmp2(lotion a,lotion b){

return a.spf<b.spf;

}

int main(){

int C,L;

scanf("%d%d",&C,&L);

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

scanf("%d%d",&cow[i].minspf,&cow[i].maxspf);

}

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

scanf("%d%d",&lot[i].spf,&lot[i].amt);

}

sort(cow,cow+C,cmp);

sort(lot,lot+L,cmp2);

int j=0;

for(int k=0;k<L;k++){

while(j<C&&lot[k].spf>=cow[j].minspf){

que.push(cow[j].maxspf);

j++;

}

while(!que.empty()&&lot[k].amt){

int temp=que.top();

que.pop();

if(temp<lot[k].spf) continue;

res++;

lot[k].amt--;

}

}

printf("%d",res);

return 0;

}

D.A data stream is a real-time, continuous, ordered sequence of items. Some examples include sensor data, Internet traffic, financial tickers, on-line auctions, and transaction logs such as Web usage logs and telephone call records. Likewise, queries over streams run continuously over a period of time and incrementally return new results as new data arrives. For example, a temperature detection system of a factory warehouse may run queries like the following. 

Query-1: "Every five minutes, retrieve the maximum temperature over the past five minutes."   
Query-2: "Return the average temperature measured on each floor over the past 10 minutes."

We have developed a Data Stream Management System called Argus, which processes the queries over the data streams. Users can register queries to the Argus. Argus will keep the queries running over the changing data and return the results to the corresponding user with the desired frequency.   
  
For the Argus, we use the following instruction to register a query: 

Register Q\_num Period

Q\_num (0 < Q\_num <= 3000) is query ID-number, and Period (0 < Period <= 3000) is the interval between two consecutive returns of the result. After Period seconds of register, the result will be returned for the first time, and after that, the result will be returned every Period seconds.   
  
Here we have several different queries registered in Argus at once. It is confirmed that all the queries have different Q\_num. Your task is to tell the first K queries to return the results. If two or more queries are to return the results at the same time, they will return the results one by one in the ascending order of Q\_num.

Input

The first part of the input are the register instructions to Argus, one instruction per line. You can assume the number of the instructions will not exceed 1000, and all these instructions are executed at the same time. This part is ended with a line of "#".   
  
The second part is your task. This part contains only one line, which is one positive integer K (<= 10000).

Output

You should output the Q\_num of the first K queries to return the results, one number per line.

Sample Input

Register 2004 200

Register 2005 300

#

5

Sample Output

2004

2005

2004

2004

2005

**代码：**

#include<iostream>

#include<cstdio>

#include<queue>

#include<functional>

#include<vector>

#include<string>

#include<algorithm>

using namespace std;

const int maxn=3000;

struct query{

int ID,P,rt;

}Q[maxn];

struct comp

{

bool operator ()(const query &a,const query &b)

{

return a.rt==b.rt?(a.ID>b.ID):(a.rt>b.rt);

}

};

priority\_queue<query, vector<query>,comp> que;

int main(){

query now;

now.rt=0;

char c[9];

while(scanf("%s",c)&&(c[0]!='#')){ //输入

//if break;

scanf("%d%d",&now.ID,&now.P);

now.rt=now.P;

que.push(now);

//i++;

}

int k;

scanf("%d",&k);

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

query temp=que.top();

que.pop();

printf("%d\n",temp.ID);

temp.rt+=temp.P;

que.push(temp);

}

query temp=que.top();

que.pop();

printf("%d",temp.ID);

return 0;

}

E. Let S = s1 s2...s2n be a well-formed string of parentheses. S can be encoded in two different ways:   
q By an integer sequence P = p1 p2...pn where pi is the number of left parentheses before the ith right parenthesis in S (P-sequence).   
q By an integer sequence W = w1 w2...wn where for each right parenthesis, say a in S, we associate an integer which is the number of right parentheses counting from the matched left parenthesis of a up to a. (W-sequence).   
  
Following is an example of the above encodings: 

S (((()()())))

P-sequence 4 5 6666

W-sequence 1 1 1456

Write a program to convert P-sequence of a well-formed string to the W-sequence of the same string.

Input

The first line of the input contains a single integer t (1 <= t <= 10), the number of test cases, followed by the input data for each test case. The first line of each test case is an integer n (1 <= n <= 20), and the second line is the P-sequence of a well-formed string. It contains n positive integers, separated with blanks, representing the P-sequence.

Output

The output file consists of exactly t lines corresponding to test cases. For each test case, the output line should contain n integers describing the W-sequence of the string corresponding to its given P-sequence.

Sample Input

2

6

4 5 6 6 6 6

9

4 6 6 6 6 8 9 9 9

Sample Output

1 1 1 4 5 6

1 1 2 4 5 1 1 3 9

代码：

#include<cstdio>

#include<cstring>

const int maxn=21;

char par[2\*maxn];

int right[maxn];

int rgl[maxn];

int res[maxn];

int main(){

int t;

scanf("%d",&t); //情况数

for(int i=1;i<=t;i++){

int n;

scanf("%d",&n);

memset(par,'(',sizeof(par)); //括号全初始为（

memset(right,0,sizeof(right)); //用于记录每个）前的（ 个数

memset(rgl,0,sizeof(rgl)); //用于记录）本身的位置

memset(res,0,sizeof(res)); //转换为w序列后的结果

for(int j=1;j<=n;j++){ //输入 每个）前的（ 个数

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

}

int lp=0,rp=0;

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

rgl[j]=right[j]+j;

par[rgl[j]]=')';

for(int k=rgl[j];k>=1;k--){

if(par[k]==')') rp++;

else if(par[k]=='(')lp++;

if(rp==lp) {res[j]=rp;break;}

}

lp=0,rp=0;

}

bool f=true;

for(int k=1;k<=n;k++){

if(f){

printf("%d",res[k]);

f=false;

}

else printf(" %d",res[k]);

}

printf("\n");

}

return 0;

}

**#include <iostream>**

**#include <stack>**

**using namespace std;**

**const int MAX = 505;**

**int fence[MAX][MAX];**

**int degree[MAX];**

**stack<int> trace;**

**int n;**

**int max(int a, int b){**

**return a>b? a: b;**

**}**

**void Eular(int k){**

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

**if(fence[k][i]){**

**fence[k][i]--;**

**fence[i][k]--;**

**Eular(i);**

**}**

**trace.push(k);**

**}**

**int main(){**

**int i,c,a,b;**

**cin>>c;**

**n = 0;**

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

**cin>>a>>b;**

**fence[a][b]++;**

**fence[b][a]++;**

**degree[a]++;**

**degree[b]++;**

**n = max(n, a);**

**n = max(n ,b);**

**}**

**bool find = false;**

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

**if(degree[i]%2 == 1){**

**find = true;**

**Eular(i);**

**break;**

**}**

**if(!find)**

**Eular(1);**

**//output**

**while(!trace.empty()){**

**cout<<trace.top()<<endl;**

**trace.pop();**

**}**

**return 0;**

**}**

A. Businesses like to have memorable telephone numbers. One way to make a telephone number memorable is to have it spell a memorable word or phrase. For example, you can call the University of Waterloo by dialing the memorable TUT-GLOP. Sometimes only part of the number is used to spell a word. When you get back to your hotel tonight you can order a pizza from Gino's by dialing 310-GINO. Another way to make a telephone number memorable is to group the digits in a memorable way. You could order your pizza from Pizza Hut by calling their ``three tens'' number 3-10-10-10.   
  
The standard form of a telephone number is seven decimal digits with a hyphen between the third and fourth digits (e.g. 888-1200). The keypad of a phone supplies the mapping of letters to numbers, as follows:   
  
A, B, and C map to 2   
D, E, and F map to 3   
G, H, and I map to 4   
J, K, and L map to 5   
M, N, and O map to 6   
P, R, and S map to 7   
T, U, and V map to 8   
W, X, and Y map to 9   
  
There is no mapping for Q or Z. Hyphens are not dialed, and can be added and removed as necessary. The standard form of TUT-GLOP is 888-4567, the standard form of 310-GINO is 310-4466, and the standard form of 3-10-10-10 is 310-1010.   
Two telephone numbers are equivalent if they have the same standard form. (They dial the same number.)   
Your company is compiling a directory of telephone numbers from local businesses. As part of the quality control process you want to check that no two (or more) businesses in the directory have the same telephone number.

Input

The input will consist of one case. The first line of the input specifies the number of telephone numbers in the directory (up to 100,000) as a positive integer alone on the line. The remaining lines list the telephone numbers in the directory, with each number alone on a line. Each telephone number consists of a string composed of decimal digits, uppercase letters (excluding Q and Z) and hyphens. Exactly seven of the characters in the string will be digits or letters.

Output

Generate a line of output for each telephone number that appears more than once in any form. The line should give the telephone number in standard form, followed by a space, followed by the number of times the telephone number appears in the directory. Arrange the output lines by telephone number in ascending lexicographical order. If there are no duplicates in the input print the line:   
No duplicates.

Sample Input

12

4873279

ITS-EASY

888-4567

3-10-10-10

888-GLOP

TUT-GLOP

967-11-11

310-GINO

F101010

888-1200

-4-8-7-3-2-7-9-

487-3279

Sample Output

310-1010 2

487-3279 4

888-4567 3

**代码：**

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

const int maxn=100010;

int st[maxn];

int change(char a){

switch (a){

case 'A' :

case 'B' :

case 'C' :

case '2' :

return 2;

break ;

case 'D' :

case 'E' :

case 'F' :

case '3' :

return 3;

break ;

case 'G' :

case 'H' :

case 'I' :

case '4' :

return 4;

break ;

case 'J' :

case 'K' :

case 'L' :

case '5' :

return 5;

break ;

case 'M' :

case 'N' :

case 'O' :

case '6' :

return 6;

break ;

case 'P' :

case 'R' :

case 'S' :

case '7' :

return 7;

break ;

case 'T' :

case 'U' :

case 'V' :

case '8' :

return 8;

break ;

case 'W' :

case 'X' :

case 'Y' :

case '9' :

return 9;

break ;

case('1'):

return 1;

break;

case('0'):

return 0;

break;

default :

break ;

}

}

bool cmp(int a,int b){

return a<b;

}

int main(){

int n;

bool t=true;

char s[210];

scanf("%d",&n);

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

st[i]=0;

scanf("%s",s);

int len=strlen(s);

for(int j=0;j<len;j++){

if(s[j]=='-')continue;

else st[i]= change(s[j])+st[i]\*10;

}

}

sort(st,st+n,cmp);

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

int ans=1;

while(st[i]==st[i+1]&&i+1<n){

t=false;

i++;

ans++;

}

if(ans>=2){

printf("%03d-%04d %d\n",st[i]/10000,st[i]%10000,ans);

}

}

if(t)printf("No duplicates.\n");

return 0;

}

B. Many databases store the data in the character fields (and especially indices) using prefix compression. This technique compresses a sequence of strings A1, ..., AN by the following method: if there are strings Ai = ai,1ai,2...ai,p and Ai + 1 = ai+1,1ai+1,2...ai+1,q   
such that for some j ≤ min(p, q) ai,1 = ai+1,1, ai,2 = ai+1,2, ... ai,j = ai+1,j, then the second string is stored as [j]ai+1,j+1ai+1,j+2... ai+1,q, where [j] is a single character with code j.   
  
If j = 0, that is, strings do not have any common prefix, then the second string is prefixed with zero byte, and so the total length actually increases.   
  
  
Constraints   
1 ≤ N ≤ 10000, 1 ≤ length(Ai) ≤ 255.

Input

First line of input contains integer number N, with following N lines containing strings A1 ... AN

Output

Output must contain a single integer -- minimal total length of compressed strings.

Sample Input

3

abc

atest

atext

Sample Output

11

代码：

#include<cstdio>

#include<cstring>

char last[260]="",now[260]="";

int n,ans,len1=0,len2=0,len=0;

int main()

{

scanf("%d",&n);

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

{

strcpy(last,now);

len1=len2;

scanf("%s",&now);

len2=strlen(now);

len=0;

for(int j=0;j<len1&&j<len2;j++)

{

if (now[j]!=last[j])

break;

len++;

}

ans+=len2-len+1;

}

printf("%d\n",ans-1);

return 0;

}

C. In this problem, you will be given one or more integers in English. Your task is to translate these numbers into their integer representation. The numbers can range from negative 999,999,999 to positive 999,999,999. The following is an exhaustive list of English words that your program must account for:   
negative, zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, hundred, thousand, million

Input

The input consists of several instances. Notes on input: 

1. Negative numbers will be preceded by the word negative.
2. The word "hundred" is not used when "thousand" could be. For example, 1500 is written "one thousand five hundred", not "fifteen hundred".

The input is terminated by an empty line.

Output

The answers are expected to be on separate lines with a newline after each.

Sample Input

six

negative seven hundred twenty nine

one million one hundred one

eight hundred fourteen thousand twenty two

Sample Output

6

-729

1000101

814022

**代码：**

#include<iostream>

#include<cstdio>

#include<cstring>

char s[50];

int sum,ans;//以百万，千，负号作为 划分区间

int change(char s[]){

if(!strcmp(s,"zero")) return 0;

else if(!strcmp(s,"one")) return 1;

else if(!strcmp(s,"two")) return 2;

else if(!strcmp(s,"three"))return 3;

else if(!strcmp(s,"four")) return 4;

else if(!strcmp(s,"five")) return 5;

else if(!strcmp(s,"six")) return 6;

else if(!strcmp(s,"seven"))return 7;

else if(!strcmp(s,"eight"))return 8;

else if(!strcmp(s,"nine"))return 9;

else if(!strcmp(s,"ten")) return 10;

else if(!strcmp(s,"eleven"))return 11;

else if(!strcmp(s,"twelve"))return 12;

else if(!strcmp(s,"thirteen"))return 13;

else if(!strcmp(s,"fourteen"))return 14;

else if(!strcmp(s,"fifteen"))return 15;

else if(!strcmp(s,"sixteen"))return 16;

else if(!strcmp(s,"seventeen"))return 17;

else if(!strcmp(s,"eighteen"))return 18;

else if(!strcmp(s,"nineteen"))return 19;

else if(!strcmp(s,"twenty"))return 20;

else if(!strcmp(s,"thirty"))return 30;

else if(!strcmp(s,"forty"))return 40;

else if(!strcmp(s,"fifty"))return 50;

else if(!strcmp(s,"sixty"))return 60;

else if(!strcmp(s,"seventy"))return 70;

else if(!strcmp(s,"eighty"))return 80;

else if(!strcmp(s,"ninety"))return 90;

}

int main(){

while(scanf("%s",s)!=EOF){

if(!strcmp(s,"negative"))printf("-");

else if((strcmp(s,"hundred"))&&(strcmp(s,"thousand"))&&(strcmp(s,"thousand"))&&(strcmp(s,"million"))){

sum+=change(s);

}

else if(!strcmp(s,"hundred")){

sum\*=100;

}

else if(!strcmp(s,"thousand")){

ans+=sum\*1000;

sum=0;

}

else if(!strcmp(s,"million")){

ans+=sum\*1000000;

sum=0;

}

// else {

// sum+=change(s);

//}

char c=getchar();

if(c=='\n'){

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

sum=0;

ans=0;

}

}

return 0;

}

D. Given two strings a and b we define a\*b to be their concatenation. For example, if a = "abc" and b = "def" then a\*b = "abcdef". If we think of concatenation as multiplication, exponentiation by a non-negative integer is defined in the normal way: a^0 = "" (the empty string) and a^(n+1) = a\*(a^n).

Input

Each test case is a line of input representing s, a string of printable characters. The length of s will be at least 1 and will not exceed 1 million characters. A line containing a period follows the last test case.

Output

For each s you should print the largest n such that s = a^n for some string a.

Sample Input

abcd

aaaa

ababab

.

Sample Output

1

4

3

**代码：**

#include<cstdio>

#include<cstring>

const int maxn=1000005;

char s[maxn];

int next[maxn];

void get\_next(char c[],int len)

{

int i=0,j=-1;

next[0]=-1;

while(i<len)

{

if(j==-1||c[i]==c[j]) {j++; i++; next[i]=j;}

else j=next[j];

}

}

int main(){

while (scanf("%s",s)){

if(s[0]=='.') break;

int len=strlen(s);

get\_next(s,len);

int n;

n=len/(len-next[len]);

if(len%(len-next[len])==0) printf("%d\n",n);

else printf("1\n");

}

return 0;

}

E. Hardwoods are the botanical group of trees that have broad leaves, produce a fruit or nut, and generally go dormant in the winter.  
America's temperate climates produce forests with hundreds of hardwood species -- trees that share certain biological characteristics. Although oak, maple and cherry all are types of hardwood trees, for example, they are different species. Together, all the hardwood species represent 40 percent of the trees in the United States.  
  
On the other hand, softwoods, or conifers, from the Latin word meaning "cone-bearing," have needles. Widely available US softwoods include cedar, fir, hemlock, pine, redwood, spruce and cypress. In a home, the softwoods are used primarily as structural lumber such as 2x4s and 2x6s, with some limited decorative applications.  
  
Using satellite imaging technology, the Department of Natural Resources has compiled an inventory of every tree standing on a particular day. You are to compute the total fraction of the tree population represented by each species.

Input

Input to your program consists of a list of the species of every tree observed by the satellite; one tree per line. No species name exceeds 30 characters. There are no more than 10,000 species and no more than 1,000,000 trees.

Output

Print the name of each species represented in the population, in alphabetical order, followed by the percentage of the population it represents, to 4 decimal places.

Sample Input

Red Alder

Ash

Aspen

Basswood

Ash

Beech

Yellow Birch

Ash

Cherry

Cottonwood

Ash

Cypress

Red Elm

Gum

Hackberry

White Oak

Hickory

Pecan

Hard Maple

White Oak

Soft Maple

Red Oak

Red Oak

White Oak

Poplan

Sassafras

Sycamore

Black Walnut

Willow

Sample Output

Ash 13.7931

Aspen 3.4483

Basswood 3.4483

Beech 3.4483

Black Walnut 3.4483

Cherry 3.4483

Cottonwood 3.4483

Cypress 3.4483

Gum 3.4483

Hackberry 3.4483

Hard Maple 3.4483

Hickory 3.4483

Pecan 3.4483

Poplan 3.4483

Red Alder 3.4483

Red Elm 3.4483

Red Oak 6.8966

Sassafras 3.4483

Soft Maple 3.4483

Sycamore 3.4483

White Oak 10.3448

Willow 3.4483

Yellow Birch 3.4483

**代码：**

#include <cstdio>

#include <iostream>

#include <map>

#include <cstring>

#include <iomanip>

using namespace std;

map<string,int>hw;

int main(){

int count=0;

string c;

while(getline(cin,c)){

hw[c]++;

count++;

}

map<string,int >::iterator i;

for(i=hw.begin();i!=hw.end();i++)

{

cout<<setiosflags(ios::fixed)<<setprecision(4)<<i->first;

cout<<" ";

cout<<100.0\*(i->second)/count<<endl;

}

return 0;

}

## （九）Training Day 9

A. A Compiler Mystery: We are given a C-language style for loop of type 

for (variable = A; variable != B; variable += C)

statement;

I.e., a loop which starts by setting variable to value A and while variable is not equal to B, repeats statement followed by increasing the variable by C. We want to know how many times does the statement get executed for particular values of A, B and C, assuming that all arithmetics is calculated in a k-bit unsigned integer type (with values 0 <= x < 2k) modulo 2k. 

Input

The input consists of several instances. Each instance is described by a single line with four integers A, B, C, k separated by a single space. The integer k (1 <= k <= 32) is the number of bits of the control variable of the loop and A, B, C (0 <= A, B, C < 2k) are the parameters of the loop.   
  
The input is finished by a line containing four zeros.

Output

The output consists of several lines corresponding to the instances on the input. The i-th line contains either the number of executions of the statement in the i-th instance (a single integer number) or the word FOREVER if the loop does not terminate.

Sample Input

3 3 2 16

3 7 2 16

7 3 2 16

3 4 2 16

0 0 0 0

Sample Output

0

2

32766

FOREVER

**代码：**

#include<cstdio>

using namespace std;

long long gcd(long long a,long long b,long long &x,long long &y){

if(b==0){

x=1;

y=0;

return a;

}

long long d=gcd(b,a%b,x,y);

long long xt=x;

x=y;

y=xt-a/b\*y; //系数x、y的取值是为满足等式d=ax+by

return d;

}

int main(){

long long A,B,C,k;

scanf("%lld%lld%lld%lld",&A,&B,&C,&k);

while(!(A==0&&B==0&&C==0&&k==0)){

long long a=C;

long long b=B-A;

long long n=(long long)1<<k; //2^k

long long x,y;

long long d=gcd(a,n,x,y);

if(b%d!=0) //方程 ax=b(mod n) 无解

printf("FOREVER\n");

else

{

x=(x\*(b/d))%n; //方程ax=b(mod n)的最小解

x=(x%(n/d)+n/d)%(n/d); //方程ax=b(mod n)的最整数小解

printf("%lld\n",x);

}

scanf("%lld%lld%lld%lld",&A,&B,&C,&k);

}

}

B. Given a *n* × *n* matrix *A* and a positive integer *k*, find the sum *S* = *A* + *A*2 + *A*3 + … + *Ak*.

Input

The input contains exactly one test case. The first line of input contains three positive integers *n* (*n* ≤ 30), *k* (*k* ≤ 109) and *m* (*m* < 104). Then follow *n* lines each containing *n* nonnegative integers below 32,768, giving *A*’s elements in row-major order.

Output

Output the elements of *S* modulo *m* in the same way as *A* is given.

Sample Input

2 2 4

0 1

1 1

Sample Output

1 2

2 3

**代码：**

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

const int maxn=31;

int n,k,mod;

struct matrix

{

int arr[maxn][maxn];

matrix()

{

memset(arr,0,sizeof(arr));

}

};

matrix multiply(matrix a,matrix b) //定义矩阵乘法

{

matrix ret;

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

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

{

if(a.arr[i][k])

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

{

ret.arr[i][j]+=a.arr[i][k]\*b.arr[k][j];

if(ret.arr[i][j]>=mod)

ret.arr[i][j]%=mod;

}

}

return ret;

}

matrix add(matrix a,matrix b) //定义矩阵加法

{

matrix an;

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

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

{

an.arr[i][j]=a.arr[i][j]+b.arr[i][j];

if(an.arr[i][j]>=mod)

an.arr[i][j]%=mod;

}

return an;

}

matrix power(matrix p,int k)

{

if(k==1) return p;

matrix e;

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

e.arr[i][i]=1;

if(k==0) return e;

while(k)

{

if(k&1)

e=multiply(p,e);

p=multiply(p,p);

k>>=1;

}

return e;

}

void output(matrix ans)

{

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

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

{

if(j==n-1)

cout<<ans.arr[i][j]<<endl;

else

cout<<ans.arr[i][j]<<" ";

}

}

matrix cal(matrix ori,int k)

{

if(k==1) return ori;

if(k&1)

return add(cal(ori,k-1),power(ori,k));//当k为奇数时，减1变为偶数 S(K)=S(K-1)+ori^K

else

return multiply(add(power(ori,0),power(ori,k>>1)),cal(ori,k>>1));

//当K为偶数时,S(K)=(1+ori^(K/2))\*S(K/2)

}

int main()

{

scanf("%d%d%d",&n,&k,&mod);

{

matrix ori,ans;

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

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

{

scanf("%d",&ori.arr[i][j]);

if(ori.arr[i][j]>=mod)

ori.arr[i][j]%=mod;

}

ans=cal(ori,k);

output(ans);

}

return 0;

}

For any even number n greater than or equal to 4, there exists at least one pair of prime numbers *p*1 and *p*2 such that

*n* = *p*1 + *p*2

This conjecture has not been proved nor refused yet. No one is sure whether this conjecture actually holds. However, one can find such a pair of prime numbers, if any, for a given even number. The problem here is to write a program that reports the number of all the pairs of prime numbers satisfying the condition in the conjecture for a given even number.

A sequence of even numbers is given as input. There can be many such numbers. Corresponding to each number, the program should output the number of pairs mentioned above. Notice that we are interested in the number of essentially different pairs and therefore you should not count (*p*1, *p*2) and (*p*2, *p*1) separately as two different pairs.

Input

An integer is given in each input line. You may assume that each integer is even, and is greater than or equal to 4 and less than 215. The end of the input is indicated by a number 0.

Output

Each output line should contain an integer number. No other characters should appear in the output.

Sample Input

6

10

12

0

Sample Output

1

2

1

代码：

#include <cstdio>

#include<cmath>

using namespace std;

bool isprime(int a){

int b=(int)sqrt(a);

for(int i=2;i<=b;i++){

int c=a/i;

int d=i\*c;

if(d==a){

return false;

}

}

return true;

}

int main(){

int t;

while(scanf("%d",&t)){

if(t==0) return 0;

int res=0;

for(int i=2,j=t-2;i<=j;i++,j--){

if(isprime(i)&&isprime(j)){

res++;

}

}

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

}

return 0;

}