> 虽然没有参加这次校赛，但 第一届 总是意义非凡的，要仔细总结！(•̀ ω •́)✧

A

1571 : 参赛人数

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

alpha想知道本次比赛的实际参赛人数。他想了个办法，看一下签到题有多少人ac，这不就行了嘛！
但是他怕签到题不简单，这样就无法统计到准确的人数。他为此很苦恼。
如果你觉得这道题很简单，也不想让alpha继续困扰下去，就输出"tjdl!"，去告诉alpha，这道题"太简单了！"

Input

无输入

Output

tjdl!

Sample Input

Sample Output

tjdl!

More Info

- **TAG：签到题**

> 别问我为什么没有题解，因为 tjdl! o(≧口≦)o

A.cpp

```cpp
#include<cstdio>
int main(){
    puts("tjdl!");
    return 0;
}
```

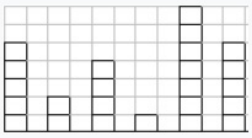
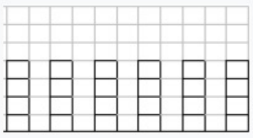
B

1572 : 积木游戏

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

beta喜欢玩积木游戏，游戏的规则是这样的：
最初有n个积木堆，各堆有自己的初始高度。游戏通过有限次数的移动，一次移动一个积木，最终使所有堆的高度相同。
beta想知道要达到目标需要的最少移动次数，他请来了精通程序设计的你来帮他算一算。

Input

输入包含多组数据，以0作为输入结束。
每组数据包含两行：
第一行为一个正整数n，表示积木堆的个数
第二行为n个正整数，h_1 h_2 ... h_n ,为每堆初始的积木个数。

数据规模约定：
$1 \le n \le 50$，$1 \le h_i \le 100$

Output

对于每一组输入数据，输出将全部堆变为相同高度最少的移动次数。
在每组输出结果之间输出一个空行。

Sample Input

```
6
5 2 4 1 7 5
2
1 3
0
```

Sample Output

```
5
1
```

More Info

在第一组样例输入中：
第1堆积木拿1个到第2堆；
第5堆积木拿1个到第2堆；
第5堆积木拿1个到第4堆；
第5堆积木拿1个到第4堆；
第6堆积木拿1个到第4堆；
即最少需5次操作。

在第二组样例输入中：
第2堆积木拿1个到第1堆；
即最少需1次操作。

- **PZ's solution:**

  1. 注意到 一次移动一块积木 ，且目标状态为 所有积木堆的积木数量 均相等，

  可以得到目标 积木堆的积木数量 一定为 $\frac{\sum_{i=1}^{n} h_i}{n}$ 即平均值；

  2. 考虑到实际操作时，比平均值多的积木堆的积木 将会移给 比平均值少的积木堆，

所以实际只有 比平均值多的积木堆 会 对答案有贡献；

3. 因为目标状态必为 所有积木堆的积木数量 均相等，答案累加的值即为 比平均值多的积木堆的 多于平均值的数量，

即对第x堆 积木数量大于平均值的积木堆，其答案贡献为 $h_x - \frac{\sum_{i=1}^{n} h_i}{n}$；

- **TAG：模拟；签到题**

B.cpp

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int n,a[55],sum,ans;
int main(){
    while(scanf("%d",&n)&&n!=0){
        sum=ans=0;
        for(int i=1;i<=n;++i) scanf("%d",&a[i]),sum+=a[i];
        sum/=n;
        for(int i=1;i<=n;++i) if(a[i]>sum) ans+=a[i]-sum;
        printf("%d\n",ans);
    }
    return 0;
}
```

C

1573 : 矩阵求和

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

我们定义这样n*n的矩阵：

```
1/1

1/1 1/2
1/2 1/1

1/1 1/2 1/3
1/2 1/1 1/2
1/3 1/2 1/1

...
```

相信你已经发现规律了：此类矩阵都是主对角线为1的对称矩阵，且主对角线两边元素的分母递增。
对于一个n*n的矩阵，请求出这个矩阵各元素的和。

Input

输入包含多组测试数据。每行给定整数n（n<50000），表示矩阵的阶数为n。
当n=0时，输入结束。

Output

对于每组输入数据，输出矩阵各元素的和，结果保留2位小数。

Sample Input

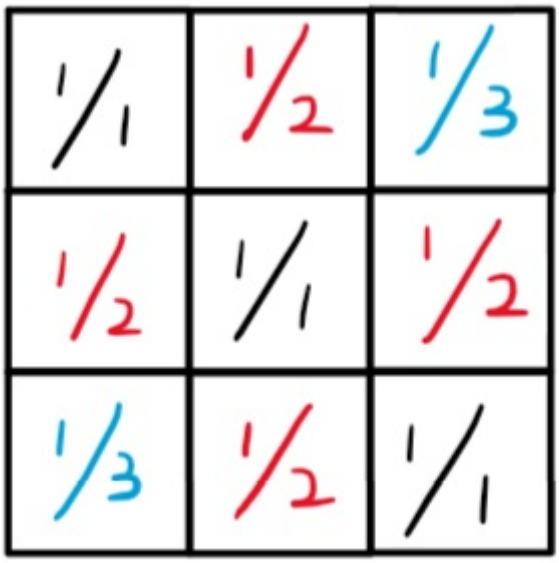
```
1
2
3
4
0
```

Sample Output

```
1.00
3.00
5.67
8.83
```

More Info

- **PZ's solution：**



1. 如图，可以发现矩阵的两个性质：

(1)矩阵内数值 沿 主对角线 （从左上角到右下角） 对称；

(2)矩阵内 数值大小 与 数值数量 呈负相关；

2.答案的计算可以从性质(2)入手，我们只观察 矩阵右上方 这一半，

显然，最小数值必为 $1/n$ ，而其数量必为 1；最大数值为 1，其数量为 n ；

3.由此我们可以大胆猜想一个关系，即 数值$1/x$的数量为$n-x+1$

- **TAG：数学；签到题**

C.cpp

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int n;
double ans;
int main(){
    while(scanf("%d",&n)&&n!=0){
        ans=0;
        for(int i=1;i<n;++i) ans+=i*1.0/(n-i+1.0);
        if(n>1) ans*=2.0;
        ans+=n;
        printf("%.2lf\n",ans);
    }
    return 0;
}
```

D

1574 : EOF

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

Rio一直觉得程序设计里有些缩写很酷。
比如End Of File可以缩写成EOF，就比eof酷。
他经过研究，发现这些各首字母大写形式的组合，其实都很酷 :)

Input

输入的第一行是一个整数T，表示一共有T组测试数据。
每组数据占一行，包含不多于20个单词，每两个单词之间由空格隔开。

规定：
每个单词长度不超过20，单词仅包含大、小写字母。

Output

每组测试数据输出其相应的缩写形式，每组输出占一行。

Sample Input

```
2
End Of File
input output system
```

Sample Output

```
EOF
IOS
```

More Info

- **TAG：模拟；签到题**

> 这道题也没有题解，因为题意本身十分明朗，有具体代码实现疑问请见D.cpp╮(╯-╰)╭

D.cpp

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
string s;
int T,n;
```

```
void print(char x){ printf("%c",'a'<=x&&x<='z' ? x-'a'+'A' : x); }
int main(){
    scanf("%d",&T);
    getline(cin,s);
    //T后有一个 '\n' 无法被scanf读取，要提前用
    //getline(cin,s); 读取消掉它!
    while(T--){
        getline(cin,s);
        n=s.size();
        print(s[0]);
        for(int i=1;i<n;++i)
            if(s[i-1]==' ') print(s[i]);
        putchar('\n');
    }
    return 0;
}
```

E

1575：整数分解

Time Limit:1.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

把一个正整数N分解成3个各不相同的正整数之和，并且要求每个正整数都不包含数字2和4，一共有多少种不同的分解方法？
注：不包含2和4指整数的各个位上的数都不为2且不为4。

Input

第一行一个整数T，表示输入有T组。
对于每一组输入数据，输入为一个正整数N。（3 < N < 2020）

Output

对每组输入数据，输出一个正整数，表示分解方法。

Sample Input

```
3
8
9
12
```

Sample Output

```
0
1
2
```

More Info

对样例的解释：
8没有满足要求的分解。
9分解为：1、3、5。
12分解为：1、3、8或1、5、6。

- **PZ's solution：**

  1.考虑到$N < 2020$，我们可以使用$O(N^2)$的算法；

  2.通过两重循环，寻找 分解成的两个数i, j，并可通过计算直接得到 第三个数为$k = n - i - j$；

  3. 我们要在 得到分解出的三个数 的判定时增加一些限制，要保证两点：

  (1)3个数 各不相同；

  (2)此答案 不能重复出现；

- **TAG：数学；签到题**

E.cpp

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
```

```cpp
#include<algorithm>
using namespace std;
int T,n,ans;
bool check(int x){
    bool f=1;
    while(x){
        if(x%10==2 || x%10==4){ f=0; break; }
        x/=10;
    }
    return f;
}
int main(){
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n);
        ans=0;
        for(int i=1;i<=n;++i)
            for(int j=1;j<i;++j){
                int k=n-i-j;
                if(i>j && j>k && k>0 &&
                   check(i) && check(j) && check(k) ) ++ans;
            }
        printf("%d\n",ans);
    }
    return 0;
}
```

F

1576：围棋程序

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

小A喜欢围棋，这天他心血来潮想要设计一个围棋小游戏，但是一个基本的问题是：该如何计算闭合线段围成的面积呢？
比如在下面这个5*5的由“0 1”组成棋盘中，由“1”围出来的“0”的面积是：3。

```
0 0 0 0 0
0 1 1 1 1
0 1 0 1 0
1 0 0 1 1
1 1 1 1 0
```

规定：面积的计算方法是统计“1”所围成的闭合曲线中水平线和垂直线交点的数目，特别地，在边上的0不被计算，如第3行第5列的“0”不算作面积。

Input

一个10*10的棋盘矩阵，棋盘中每个点由0、1组成。

Output

由题意输出1围成的0的面积。

Sample Input

```
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
0 0 0 1 1 1 0 0 0 0
0 0 0 1 0 0 1 1 1 0
0 0 0 0 1 1 0 0 0 1
0 0 0 0 0 1 0 0 1 0
0 0 0 1 1 0 0 1 0 0
0 0 0 1 0 0 0 1 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 1 1 1 0 0 0
```

Sample Output

```
12
```

More Info

- **PZ's solution：**

  1.考虑使用 BFS ，判断0是否 可能 为被围面积的起点，可以通过判断 其上 和 其左 是否为1;

  2.在BFS过程中，可以发现，不被围起来的0 一定会遍历到边界，通过此特点判断 无效起点

  3.实现和细节请见代码(¬‿¬)

- **TAG：BFS广度优先搜索**

F.cpp

```cpp
#include<iostream>
#include<algorithm>
```

```cpp
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
queue<int>qx,qy;
int fx[]={0,0,1,-1};
int fy[]={1,-1,0,0};
int a[15][15],nx,ny,res,ans;
bool vis[15][15],f;
int  bfs(int x,int y){
    res=0; f=0;
    qx.push(x); qy.push(y); vis[x][y]=1; ++res;
    while(!qx.empty()){
        x=qx.front(); qx.pop();
        y=qy.front(); qy.pop();
        for(int i=0;i<4;++i){
            nx=x+fx[i]; ny=y+fy[i];
            if(vis[nx][ny]||a[nx][ny]==1) continue;
            if((nx==1||nx==10||ny==1||ny==10)&&a[nx][ny]==0){ res=0; f=1; }
            //虽然此处可以判断 非法0起点，但仍让其遍历完所有的0
            //因为我们设置了 vis[x][y] 来表示已经遍历过的点，如果出现 非法边界被遍历过的情况，就可能无法再 判断非法0起点 了
            if(a[nx][ny]==0&&!vis[nx][ny]&&2<=nx&&nx<=9&&2<=ny&&ny<=9){
                if(!f) ++res;
                vis[nx][ny]=1;
                qx.push(nx); qy.push(ny);
            }
        }
    }
    return res;
}
int main(){
    for(int i=1;i<=10;++i)
        for(int j=1;j<=10;++j)
            scanf("%d",&a[i][j]);
    for(int i=2;i<=9;++i)
        for(int j=2;j<=9;++j)
            if(a[i][j]==0&&a[i-1][j]==1&&a[i][j-1]==1&&!vis[i][j])
                ans+=bfs(i,j);
    printf("%d",ans);
    return 0;
}
```

G

1577：搬家

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

要说我校的搬家史那可有的说的了。就alpha来说，他这个暑假刚经历一次搬寝室。
alpha寝室里的东西有n件，但搬东西真的是件很累的事，他只准备随便搬2*m件就行。
alpha每趟左手、右手分别拎一件物品。
凭借天资聪颖，他迅速地发现了：每搬一趟，他的疲劳度与左、右手的物品的重量差的平方成正比。例如他左手拿重量为3的物品,右手拿重量为6的物品,则他搬完这次的疲劳度为(6-3)^2 = 9.
他想知道他搬完这2*m件物品后疲劳度最低是多少？

Input

每组输入数据有两行。
第一行有两个数n,m (2*m ≤ n < 2000)。
第二行有n个整数分别表示n件物品的重量(重量是一个小于2^{15}的正整数)。
0 0表示输入结束。

Output

对于每组输入数据，输出一个正整数，表示他最少的疲劳度，每组输出占一行。

Sample Input

```
5 1
18467 6334 26500 19169 15724
7 1
29358 26962 24464 5705 28145 23281 16827
0 0
```

Sample Output

```
492804
1399489
```

More Info

- **PZ's solution：**

  1. 首先考虑贪心，对物品的重量a_i进行排序；

  2. 显然， 答案贡献必为 $(a_i - a_{i-1})^2$ 的形式，即相邻两数的平方差；

  3. 但此时，贪心思想到此结束，因为我们注意到，有一种情况可能存在；

  ！：我们选择 最小的 $a_i - a_{i-1}$ 后，导致 $a_{i+1} - a_{i-2}$ 及 所有其他答案贡献 极大，造成答案非最优

  4.考虑动态规划，设 $f[i][j]$ 表示 前i个物品 选了j对搬走 时的最优答案，

  $f[i][j]$可由两个状态转移过来：

  (1)选择搬走a_i、a_{i-1}，此时$f[i][j] = f[i-2][j-1] + (a_i - a_{i-1})^2$

  (2)选择不搬a_i，此时$f[i][j] = f[i-1][j]$

  有状态转移方程

  $f[i][j] = min(f[i-2][j-1] + (a_i - ai-1)^2, f[i-1][j])$

- **TAG：贪心；DP动态规划**

G.cpp

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<climits>
using namespace std;
int n,m,a[2005],f[2005][2005];
int main(){
    while(scanf("%d %d",&n,&m)&&n!=0){
        memset(f,0x3f,sizeof(f));
        for(int i=1;i<=n;++i) scanf("%d",&a[i]);
        sort(a+1,a+1+n);
        for(int i=0;i<=n;++i) f[i][0]=0;
        //初始化可转移的合法状态，即 一对都不选 的状态
        for(int i=2;i<=n;++i)
            for(int j=1;j<=min(m,i/2);++j)
            //考虑边界，当选择到i时，最多可以选 i/2 对物品,而题目同时也限制最多选m对物品
                f[i][j]=min(f[i-2][j-1]+(a[i]-a[i-1])*(a[i]-a[i-1]),f[i-1][j]);
        printf("%d\n",f[n][m]);
    }
    return 0;
}
```

H

1578 : 社团活动

Time Limit:2.000 Sec Memory Limit:128 MB
Problem Tags: 第一届ACM校赛 比赛 Manager:zhbink

Description

bbq的工作是管理学校的社团活动，具体来说是为每个社团活动分配教室。要把有限的教室合理安排给这些社团，是不容易的。
每个社团活动用k，t1，t2来表示：该社团活动在第t1天～第t2天内需要k个教室（包括t1，t2）。
bbq总是按社团活动申请的先后顺序分配教室，如果某一天剩余的教室数量不够满足某社团的要求，则停止教室的分配。bbq需要告知该社团，他们的该次社团活动无法进行。

Input

每组输入数据第一行包括两个正整数n、m，为总天数和社团活动的总数量。
第二行包含n个正整数，其中第i个数为r_i，表示第i天空教室的数量。
接下来有m行，每行为一个社团活动的信息，包含三个正整数k，t_1，t_2。（k，t1，t2如题目描述）。

规定：
天数与社团活动编号均用从1开始的整数编号。
$1 \le n, m \le 10^6$，$0 \le ri \le 10^9$，$0 \le k \le 10^9$ ，$1 \le t_1 \le t_2 \le n$。

Output

如果所有社团的申请均可满足，则输出0。
否则输出一个正整数，为需要bbq告知的活动无法进行的社团活动编号。

Sample Input

```
4 3
2 5 4 3
2 1 3
3 2 4
4 2 4
```

Sample Output

```
2
```

More Info

- **PZ's solution：**

  1.根据题意，直接考虑 (。·∀·)ﾉ゙线段树维护区间最小值；

  2.区间操作 即可视为 区间减 操作；

  > 其他做法请大家自己探索吧，我会将标程做法也贴在下面

- **TAG：线段树；差分；二分答案**

H.cpp

```cpp
#include<iostream>
#include<algorithm>
#include<cstdio>
#include<cstring>
```

```cpp
using namespace std;
#define mid (l+r>>1)
#define lo o<<1
#define ro o<<1|1
#define N 1000005
int minx[N<<2],lzy[N<<2],n,m,k,t1,t2,ans;
bool f;
void build(int l,int r,int o){
    if(l==r){ scanf("%d",&minx[o]); return; }
    build(l,mid,lo); build(mid+1,r,ro);
    minx[o]=min(minx[lo],minx[ro]);
}
void pushdown(int o){
    minx[lo]-=lzy[o];
    minx[ro]-=lzy[o];
    lzy[lo]+=lzy[o];
    lzy[ro]+=lzy[o];
    lzy[o]=0;
}
void updata(int l,int r,int L,int R,int k,int o){
    if(lzy[o]) pushdown(o);
    if(f) return;
    if(L<=l&&r<=R){
        if(minx[o]<k){ f=1; return; }
        minx[o]-=k;
        lzy[o]+=k;
        return;
    }
    if(L<=mid) updata(l,mid,L,R,k,lo);
    if(R>mid) updata(mid+1,r,L,R,k,ro);
    minx[o]=min(minx[lo],minx[ro]);
}
int main(){
    scanf("%d %d",&n,&m);
    build(1,n,1);
    for(int i=1;i<=m;++i){
        scanf("%d %d %d",&k,&t1,&t2);
        if(!f){
            updata(1,n,t1,t2,k,1);
            if(f) ans=i;
        }
    }
    printf("%d",ans);
    return 0;
}
```

std.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6+50;
using ll = long long;
int n,m;
ll a[N],room[N],sum[N],t1[N],t2[N];
```

```cpp

bool check(int mid)
{
    memset(sum,0,sizeof(sum));
    for (int i=1;i<=mid;i++)
    {
        sum[t1[i]]-=room[i];
        sum[t2[i]+1]+=room[i];
    }

    int cnt=0;
    for (int i=1;i<=m;i++)
    {
        cnt+=sum[i];
        if (a[i]+cnt < 0) return 0;
    }
    return 1;
}
int main()
{
    scanf("%d%d",&n,&m);
    for (int i = 1; i <= n; i++)
        scanf("%lld", &a[i]);
    for (int i = 1; i <= m; i++)
        scanf("%lld%lld%lld", &room[i],&t1[i],&t2[i]);
    int l=1, r=m+1;
    while(l < r)
    {
        int mid = (l + r) >> 1;
        if (check(mid)) l = mid+1;
        else r = mid;
    }
    if (l == m+1) {printf("0\n");return 0;}
    printf("%d\n",l);
    return 0;
}
```

吐槽

1.这次校赛难度不难，共8道题，有5道签到题（在我看来，只要能第一眼看出 做法和细节 的题，都算签到题(*￣3￣)╭）；

2.自己做题的过程中，F题没有注意细节，G题一路贪心到底一直错，H题因为`build(1,1,n)`的粗心错误一直错，发现自己也是菜的要死。

3.这次比赛的学长我只认识一位ssw，毕竟自己刚来到这里，让我们第二次校赛的题解见，到时候就有自己赛场感受了！