Problem A

=========

类似超市结账，计算购买的商品的总价格。

输入

第一行为测试数据组数T（0<T<=10）

每组数据第一行为购买商品的种类n，接下来n行，每行两个数据，第一个为商品价格，第二个为商品数量，价格为实型。

## 输出

每一行输出相对应数据的总价值，保留两位小数。

## 样例输入

2  
2  
1.00 2  
0.50 2  
1  
100.0 1

## 样例输出

3.00  
100.00

## 代码

#include <bits/stdc++.h> \#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int main()  
{  
 freopen("in","r",stdin);  
 int T,N;  
  
 scanf("%d",&T);  
 while(T--){  
 double sum=0;  
 double t;  
 int n;  
 scanf("%d",&N);  
 while(N--){  
 scanf("%lf%d",&t,&n);  
 sum+=t\*n;  
 }  
 printf("%.2f\n",sum);  
 }  
 return 0;  
}

# ProblemB

V字型数列，当且仅当三元组i<j<k，且 a[i]>a[j]并且a[k]>a[j]，算作一个V型数列。

输入

第一行为测试数据组数T

每组数据第一行为该数组的数字个数

接下来一行为数组元素

输出

输出对应数组中符合v字形数列的个数

测试数据

2  
3  
2 1 2  
5  
2 1 2 1 2

输出

1  
4

代码

复杂度比较高，待优化

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int a[LEN];  
  
int main()  
{  
 freopen("in","r",stdin);  
 int T,N;  
 scanf("%d",&T);  
 while(T--){  
 scanf("%d",&N);  
 FF(i,N)  
 scanf("%d",&a[i]);  
 int ans=0;  
 for(int i=1;i<N-1;i++){  
 int n=0,m=0;  
 for(int j=0;j<i;j++){  
 if(a[j]>a[i]){  
 n++;  
 }  
 }  
 for(int j=i+1;j<N;j++){  
 if(a[j]>a[i])  
 m++;  
 }  
 ans+=n\*m;  
 }  
 printf("%d\n",ans);  
 }  
 return 0;  
}

Problem C. 24点

===============

题目描述

输入为4个数字，a,b,c,d。若abcd=24，则称该数组满足24点，其中可以为+、-、\*、/任一个，若不能得到结果24，则称该数组不满足24点。（计算为实型）

输入

第一行为测试组数t

接下来t行，每一行为四个整数a,b,c,d，测试是否满足24点

1<=a,b,c,d<10000（右区间忘了，反正四个数不为0）

输出

在每一行

若满足，输出YES

不满足，输出NO

样例

* 输入

3  
2 6 3 4  
1 1 2 3  
3 2 2 8

* 输出

// 2+6\*3+4  
Yes  
// 1 1 2 3  
No  
// 3\*2/2\*8  
Yes

代码

* 思路：

1. 先通过笛卡尔积（product），对于4个操作符生成P（4,3）的穷举序列。
2. 对于每一组测试数据，把穷举的操作符序列与操作数带入==judge==函数进行判断
3. 在==judge==函数中，按照算符优先级进行运算。

* 拓展

1. 笛卡尔积采用递归进行构造。如果考场上想不出，可以使用3重循环。
2. 表达式计算采用简化的模式。如果需要像Python的eval一样计算，参照：

* https://blog.csdn.net/TQCAI666/article/details/88692096

虽然没有OJ可以提交，但是我想记录一下坑，那就是**最好用浮点数进行计算**

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int num[4];  
string a="+-\*/";//递归前  
vector<string> ans;//递归结果  
  
string ch2str(char fuck){  
 char fuckYou[2]={0};  
 fuckYou[0]=fuck;  
 return fuckYou;  
}  
  
void product(int t,int n,int m){//P(n,m)  
 if(t>=m){//达到循环上限  
 return;  
 }else{  
 vector<string> tmp;  
 int sz=ans.size();  
 if(sz){  
 for(int i=0;i<n;i++){  
 for(int j=0;j<sz;j++){  
 tmp.push\_back(ans[j]+a[i]);  
 }  
 }  
 }else{  
 for(int i=0;i<n;i++){  
 tmp.push\_back(ch2str(a[i]));  
 }  
 }  
 ans=tmp;  
 product(t+1,n,m);  
 }  
}  
  
int operate(int &a,int& b,char op){  
 int v;  
 switch(op){  
 case '+':  
 v=a+b;  
 break;  
 case '-':  
 v=a-b;  
 break;  
 case '\*':  
 v=a\*b;  
 break;  
 case '/':  
 v=a/b;  
 break;  
 }  
 return v;  
}  
  
bool judge(string op,int\*num){  
 string opbk=op;  
 string p="\*/+-";//优先级列表  
 vector<int> v(num,num+4);  
 while(op.size()>0){  
 FF(i,4){//按照运算符优先级进行遍历  
 FF(j,op.size()){  
 if(p[i]==op[j]){  
 int ans=operate(v[j],v[j+1],op[j]);  
 v[j]=ans;  
 v.erase(v.begin()+j+1);  
 op.erase(j,1);  
 break;  
 }  
 }  
 }  
 }  
 //测试代码  
 if(v[0]==24){  
 FF(i,3){  
 cout<<num[i];  
 cout<<opbk[i];  
  
 }  
 cout<<num[3]<<endl;  
 }//end  
 return v[0]==24;  
  
}  
  
int main()  
{  
 freopen("in","r",stdin);  
 int T,N;  
 scanf("%d",&T);  
 product(0,4,3);//完成之后，行程的就是操作符表ans  
 while(T--){  
 FF(i,4){  
 scanf("%d",&num[i]);  
 }  
 bool yes=0;  
 for(int i=0;i<ans.size();i++){//对操作符序列进行穷举  
 string op=ans[i];  
 bool pd=judge(op,num);  
 if(pd){  
 yes=1;  
 break;  
 }  
 }  
 puts(yes?"Yes":"No");  
 }  
 product(0,4,3);  
 return 0;  
}

Problem D. 最大价值

===================

对于每一个零件都有一个适宜温度区间，[Ri,Ji]，当温度t<Ri，零件价值为x，当t>Ji，零件价值为z；当温度适宜，价值为y。且y>x,y>z。此刻，有一恒温箱，可确定温度t。

输入

第一行按顺序分别为 n,x,y,z。0<n<20000

接下来n行为 每一个零件的适宜温度区间0<Ri<Ji<109

输出

确定一个温度值t，在这温度下所有零件的总价值最大，并输出该价值

测试数据

3 1 3 2  
1 4  
2 5  
7 10

输出

7

代码

能过样例，但不确定是否能AC 复杂度：Nlog2N N∈[1,20000)

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int main()  
{  
 vector<int> left;//左跨边  
 vector<int> right;//右跨边  
 vector<int> node;//计算时的结点  
 freopen("in","r",stdin);  
 int N,x,y,z,l,r;//0 < n <20000  
 scanf("%d%d%d%d",&N,&x,&y,&z);  
 FF(i,N){  
 scanf("%d%d",&l,&r);  
 left.push\_back(l);  
 right.push\_back(r);  
 node.push\_back(l+1);//选取中间为采样点  
 node.push\_back(r-1);  
 }  
 sort(left.begin(),left.end());  
 sort(right.begin(),right.end());  
 sort(node.begin(),node.end());  
 int ans=0;  
 FF(i,node.size()){  
 int cur=node[i];  
 vector<int>::iterator it;  
 //右比当前小  
 it=upper\_bound(right.begin(),right.end(),cur);  
 int lCnt=distance(right.begin(),it);  
 //左比当前大  
 it=lower\_bound(left.begin(),left.end(),cur);  
 int rCnt=distance(it,left.end());  
 int cCnt=N-lCnt-rCnt;  
 int sum=rCnt\*x+cCnt\*y+lCnt\*z;  
 ans=max(ans,sum);  
 }  
 printf("%d\n",ans);  
 return 0;  
}