小型Matlab的实现方式

这个页面用于吐槽多校第四场的一道题目。

作为面向对象Java大作业感觉不错，建议未来的助教们可以考虑考虑。

原型GitHub项目找到了……好像是python写的一个大项目，叫geosolver

GitHub项目链接：geosolver

引言与题意

ZYB对数学有敏锐的直觉，尤其是在几何问题上。

几何问题是这样的：求∠CAM的值。

或者是这样的：如果AC=x−3，BE=20，AB=16，CD=x+5，求x。

为了更容易地分析问题，输入将包含逻辑形式，而不是原始的问题文本和图表。

给出若干个符合一定规则的几何题条件。

要求出某个角或者某条边或未知数x的值。

基本逻辑形式

- 数字。使用十进制整数表示数字。
- 未知数字x是唯一未知的数字。
- 表达式Expression：表达式可以是一个数字，也可以是一个表达式，其中x只出现一次，最多一次加减法，最多一次乘法。乘法符号可以省略。
  - 例如：233，3x+5，x*2+3，x-2是有效表达式，但3x+5-3，x+2x，5*3，2y不是。
- 点。使用单大写字母表示点。
- 线。用Line(Point, Point)来表示一条线（实际上它是一条线段）。
  - 例如：Line(A, B)。
- 角。使用Angle(Point, Point, Point)来表示一个角。
  - 例如：Angle(A,B,C)。
- 圆。使用Circle(Point)表示具有特定中心的圆。
  - 例如：Circle(O)。
- 线段长。使用LengthOf(Line)来获得特定线段的长度值。
  - 例如：LengthOf(Line(A, B))。
- 角度。用MeasureOf(Angle)得到特定角度的度数值。
  - 例如：MeasureOf(Angle(A,B,C))。
- 项Term：项=线段长|角度|表达式。
- 相等。使用Equals(Term,Term)来声明这两个项的值相等。
  - 例如：Equals(LengthOf(A,B), 2)，Equals(MeasureOf(Angle(A, B, C))。
- 垂直。使用Perpendicular(Line, Line)表示两条垂直线。
  - 例如：Perpendicular(Line (A, C), Line(B, D))。
- 平行。使用Parallel(Line,Line)表示两条平行线。保证各点都是有序的。

    - 例如：Parallel(Line (A, C), Line(B, D))。
- 点在线上。使用PointLiesOnLine(Point, Line)来表示位于直线上的点。
    - 例如：PointLiesOnLine(A, Line(B, C))。
- 点在圆上。使用PointLiesOnCircle(Point, Circle)表示位于圆上的点。
    - 例如：PointLiesOnCircle(A, Circle(O))。
- 问题。使用Find(Term)来询问给定项的确切值。
    - 例如：Find(x)，Find(LengthOf(Line(A,B)))。

请注意，图和文本中的所有条件都将转换为逻辑形式。你现在得到了一个只有一个问题（Find phrase）的逻辑表单列表，并希望找到解决方案。

定理

- 平角定理：如果点C位于AB上，则∠ACB=180∘。
- 等腰三角形定理：在三角形ABC中，如果AB=AC，则∠ACB=∠ABC，反之亦然。
- 三角形内角和定理：任何三角形的三个内角加起来等于180 。
- 勾股定理：在直角三角形中，如果三条边的长度分别是a，b，c（其中c是斜边），那么a^2+b^2=c^2。
    - 这个定理的逆定理也成立，但是暂时用不到。
- 全等三角形定理：两个三角形满足下列条件时为全等。
    - SSS（如果三条边对应相等，则两个三角形全等。
    - SAS（如果两边及其夹角对应相等，则两个三角形全等。
    - AAS和ASA（如果有两个对应角相等（相似），又有一组对应边相等，则两个三角形全等。
    - HL（对应斜边和对应直角边相等的两个直角三角形全等。
    - 在全等三角形中，对应边和对应角相等。
- 相似三角形定理：两个三角形在满足以下条件时是相似的。
    - SSS（三条对应边成比例的两个三角形相似。
    - SAS（两条对应边成比例，且夹角相等，则两个三角形相似。
    - AA（两个对应角相等的两个三角形相似。
    - HL（对应斜边和对应直角边成比例的两个直角三角形相似。
    - 在相似三角形中，对应角相等，对应边成比例。
- 平行线定理：如果两条直线平行，则同位角（corresponding）相等，内错角（alternate）相等，同旁内角（interior）互补（supplementary）。
- 直径相等定理：同一个圆的不同直径相等。圆的中心也是每个直径的中点。
- 圆上点定理：如果AB是圆O的直径，另一个点C位于圆O上，则∠ACB=90∘。

建议

- 逻辑形式可以嵌套。
- 保证每个案例至少有一个好的解决方案。一个好的解决方法是：你可以用上面的定理一步一步地解决问题；每一步之后，你获得的新值都是有效表达式（这也意味着表达式中涉及的数字总是整数）；步骤数不超过4。
- 给定的逻辑形式是充分的。
    - 例如：如果一个段上有四个点A，B，C，D，那么将列出四个相应的PointLiesOnLine短语（A-B-C，A-B-D，A-C-D，B-C-D）。
    - 例如：如果给你一个短语Perpendicular(Line (A, C), Line(B, D))，这两条线的交集也会给出。
        - 即，PointLiesOnLine(E, Line(A, C))和PointLiesOnLine(E, Line(B, D))，如果交集是E。
- 如果AO，BO，CO是三个不同的线段，则很难检测哪个线段位于中间。在这个问题中，您不需要

在∠AOB□∠AOC□∠BOC之间建立关系。
    - 除非在这种明显的情况下：如果B位于AC上，则可以使用∠AOC=∠AOB+∠BOC□
- 所有的数据都来自现实世界的问题，而不是人工构建的。
- 总共有20个问题。从中选取样本（包括10个问题）。

输入输出描述

输入包含多个样例。输入的第一行包含一个整数T□总是5），即样例数。

对于每个样例，第一行包含一个整数N□3≤N≤12□□表示逻辑形式的数量。第二行包含几个以空格分隔的大写字母，表示与此问题相关的所有点。第三行包含几个长度为2的大写字母字符串，表示与此问题相关的所有直线（线段）。在接下来的N行中，有一个字符串si□|si|≤50□□表示第i个逻辑形式。

保证最后一个逻辑形式以“Find”开头。

对于每种情况，输出一个表示答案的整数。不应该牵涉到未知的数字。

样例一

```
5
11
N M B C T A D
MA MB MT MD AB AC BC TC TN CN
Equals(MeasureOf(Angle(M, T, N)), 28)
PointLiesOnLine(B, Line(A, C))
PointLiesOnLine(D, Line(T, N))
PointLiesOnCircle(N, Circle(M))
PointLiesOnCircle(C, Circle(M))
PointLiesOnCircle(T, Circle(M))
PointLiesOnCircle(A, Circle(M))
Perpendicular(Line(A, B), Line(M, B))
Perpendicular(Line(D, M), Line(T, D))
Equals(LengthOf(Line(B, M)), LengthOf(Line(D, M)))
Find(MeasureOf(Angle(C, A, M)))
8
A B C D E
AC AD AB AE BC BE CD DE
Equals(LengthOf(Line(A, C)), x-3)
Equals(LengthOf(Line(A, B)), 16)
Equals(LengthOf(Line(C, D)), x+5)
Equals(LengthOf(Line(B, E)), 20)
PointLiesOnLine(C, Line(A, D))
PointLiesOnLine(B, Line(A, E))
Parallel(Line(B, C), Line(E, D))
Find(x)
12
A B C D F
CB CD CA CF BD BA BF DA DF AF
Equals(MeasureOf(Angle(F, A, D)), 20)
```

```
Equals(LengthOf(Line(D, A)), 9)
Equals(MeasureOf(Angle(F, A, B)), 32)
Equals(LengthOf(Line(B, A)), 6)
Equals(MeasureOf(Angle(A, D, B)), 40)
PointLiesOnLine(F, Line(C, A))
PointLiesOnLine(F, Line(B, D))
Parallel(Line(A, D), Line(B, C))
Equals(LengthOf(Line(A, D)), LengthOf(Line(B, C)))
Parallel(Line(A, B), Line(D, C))
Equals(LengthOf(Line(A, B)), LengthOf(Line(D, C)))
Find(MeasureOf(Angle(D, B, A)))
5
A B C
AB BC AC
Equals(LengthOf(Line(A, B)), 2x-7)
Equals(LengthOf(Line(B, C)), 4x-21)
Equals(LengthOf(Line(A, C)), x-3)
Equals(LengthOf(Line(A, B)), LengthOf(Line(B, C)))
Find(LengthOf(Line(A, C)))
5
A B C
AB AC BC
Equals(LengthOf(Line(A, C)), 3)
Equals(LengthOf(Line(A, B)), 5)
Equals(LengthOf(Line(B, C)), x)
Perpendicular(Line(A, C), Line(B, C))
Find(x)

28
35
88
4
4
```

样例二

```
5
7
A C B D E
AB AC AE AD BE BC DE CD
Equals(LengthOf(Line(A, C)), 16)
Equals(LengthOf(Line(E, D)), 5)
Equals(LengthOf(Line(A, B)), 12)
PointLiesOnLine(B, Line(A, C))
Parallel(Line(C, D), Line(B, E))
PointLiesOnLine(E, Line(A, D))
Find(LengthOf(Line(A, E)))
12
```

```
A B C D F
CB CD CA CF BD BA BF DA DF AF
Equals(MeasureOf(Angle(F, A, D)), 20)
Equals(LengthOf(Line(D, A)), 9)
Equals(MeasureOf(Angle(F, A, B)), 32)
Equals(LengthOf(Line(B, A)), 6)
Equals(MeasureOf(Angle(D, B, C)), 40)
PointLiesOnLine(F, Line(C, A))
PointLiesOnLine(F, Line(B, D))
Parallel(Line(A, D), Line(B, C))
Equals(LengthOf(Line(A, D)), LengthOf(Line(B, C)))
Parallel(Line(A, B), Line(D, C))
Equals(LengthOf(Line(A, B)), LengthOf(Line(D, C)))
Find(MeasureOf(Angle(A, D, C)))
12
A B C D E F G
GC GD GB GF GA GE CE BF BA FA
Equals(MeasureOf(Angle(A, G, C)), 60)
PointLiesOnLine(F, Line(G, A))
PointLiesOnLine(G, Line(C, E))
PointLiesOnLine(G, Line(B, F))
PointLiesOnLine(G, Line(B, A))
PointLiesOnLine(F, Line(B, A))
PointLiesOnCircle(C, Circle(G))
PointLiesOnCircle(B, Circle(G))
PointLiesOnCircle(A, Circle(G))
PointLiesOnCircle(E, Circle(G))
Perpendicular(Line(G, F), Line(G, D))
Find(MeasureOf(Angle(B, G, E)))
3
A B C
AB AC BC
Equals(MeasureOf(Angle(A, B, C)), 40)
Equals(MeasureOf(Angle(C, A, B)), 25)
Find(MeasureOf(Angle(B, C, A)))
6
D A B K G
KG GD DA KA AB KB
Equals(MeasureOf(Angle(B, A, D)), 3x-70)
Equals(MeasureOf(Angle(K, G, D)), 120)
Equals(MeasureOf(Angle(G, D, A)), x)
Parallel(Line(K, G), Line(A, D))
PointLiesOnLine(A, Line(K, B))
Find(x)

15
128
60
115
60
```

核心思路

题目保证了每一步的结果依然是 expressions

我们不断用已知的条件和定理扩展我们知道的量，直到找到答案。中途可能需要构方程和解方程。

整个过程有点像迭代加深搜索。

注意点

同一个角的表示有很多种，注意要对它们建立Equal关系。

对于所有 PointLiesOnLine要构建边长相加的 Equal 关系和角度相加的 Equal 关系。

对于平行线，同位角可能不存在，所以可以只用内错角和同旁内角建立关系。

可以先不断地用“ 基本定理 ”传播 expression到最后再用勾股定理、相似定理等复杂的定理解方程。

要能检测出图中的三角形（只要三点不共线就算），并积极寻找全等和相似的三角形对。

在运用一些定理时，所用的条件可能不是“ 完全 ”的。比如我们知道两条边的长度都是 2x+3我们依然可以利用它们相等来构建 对角相等 或者 三角形全等 的关系。

可以用 Find 导向去优化搜索方向。不过这道题是不必要的，你把所有可以求的量求出来也是 OK 的。

在解出 x 后，要把之前所有用 x 表示的表达式都带入一遍。

复杂度为 所有状态量 * 定理数量 * 步数上限（4）。

目前的代码

其中auto关键字需要C++11才能通过编译。

目前情况是“ 运行错误 ”。这是因为在代码末尾有一个throw代码可以通过两个给出样例，但是如果注释掉throw会“ 答案错误 ”，只能通过50%的样例。

代码中还有两个被注释掉的throw去掉注释完全不影响，说明可能不是读入与判断相等层次的问题，可能是其他逻辑问题。建议参考一下命题人的其他提示，并检查代码的相应部分是否按照出题人提示完成了。

```
#include<stdio.h>
#include<math.h>
#include<string.h>

#include<iostream>
#include<sstream>
#include<algorithm>

#include<set>
```

```
#include<map>
#include<queue>
#include<string>
#include<bitset>
#include<functional>
#include<random>

using namespace std;

#define REP(_i,_a,_n) for(int _i=_a;_i<=_n;++_i)

int Value=1e8;

vector<char> BasePoint;
struct Expression {
    int x,y;
    Expression () {}
    Expression (int x, int y) :x(x),y(y) {}
    bool operator == (const Expression & b) const {return x==b.x&&y==b.y;}
    bool operator != (const Expression & b) const {return x!=b.x||y!=b.y;}
    Expression operator + (const Expression & b) {
        return Expression(x+b.x,y+b.y);
    }
    Expression &operator += (const Expression & b) {
        return *this=*this+b;
    }
    Expression operator - (const Expression & b) {
        return Expression(x-b.x,y-b.y);
    }
    Expression &operator -= (const Expression & b) {
        return *this=*this-b;
    }
};
struct Line {
    char x,y;
    Line (char x=0, char y=0) :x(x),y(y) {}
    Line repr() const {
        Line u = *this;
        if (u.x>u.y) swap(u.x,u.y);
        return u;
    }
    bool operator == (const Line &b) const {return x==b.x&&y==b.y;}
    bool operator != (const Line &b) const {return x!=b.x||y!=b.y;}
    bool operator < (const Line &b) const {
        Line u = this->repr();
        Line v = b.repr();
        if (u.x!=v.x) return u.x<v.x;
        return u.y<v.y;
    }
} sourceLine[5555];
map<Line,int> BaseLine;
```

```
map<Line,int> length;
map<Line,Expression> length2;
map<pair<Line,Line>,int> parallel;
map<pair<Line,Line>,int> perpendicular;
map<Line,vector<char> > Point_Line;
struct Angle {
    char x,y,z;
    Angle (char x=0, char y=0, char z=0) :x(x),y(y),z(z) {}
    Angle repr() const {
        Angle u = *this;
        if (u.x>u.z) swap(u.x,u.z);
        return u;
    }
    bool operator < (const Angle & b) const {
        Angle u = this->repr();
        Angle v = b.repr();
        if (u.x!=v.x) return u.x<v.x;
        if (u.y!=v.y) return u.y<v.y;
        return u.z<v.z;
    }
    void pr() {
        //printf("%c%c%c",x.x,y.x,z.x);
    }
} sourceAngle[5555];
map<Angle,int> degree;
map<Angle,Expression> degree2;
map<Angle,int> BaseAngle;
struct Triangle {
    char x,y,z;
    Triangle (char x=0, char y=0, char z=0) :x(x),y(y),z(z) {}
    bool operator < (const Triangle & b) const {
        if (x!=b.x) return x<b.x;
        if (y!=b.y) return y<b.y;
        return z<b.z;
    }
} sourceTriangle[5555];
map<Triangle,int> BaseTriangle;

map<char,vector<char> > Point_Circle;
struct Term {
    int type;
    Line l;
    Angle ang;
    Expression val;
} Question;
struct Line_Equation {
    int var1, var2, var3, var4;
    Expression val;
};
int Line_fa[5555];
```

```
int Angle_fa[5555];
int Find(int fa[5555], int x) {return fa[x]==x?x:fa[x]=Find(fa,fa[x]);}
//0, l1=l2
//1, l1+l2=l3
//2, l1^2+l2^2=l3^2
//3, l1/l2=l3/l4
vector<Line_Equation> Base_Line_Equation[4];
struct Angle_Equation {
    int var1, var2, var3;
};
//0, a1=a2
//1, a1+a2=180
//2, a1+a2=a3
//3, a1+a2+a3=180
vector<Angle_Equation> Base_Angle_Equation[4];
struct Logic_Form {
    char p;
    Line l;
    Angle ang;
    char c;
    Term t;
};
int CreatePoint(char x) {
    for (auto &y:BasePoint) if (y==x) return 0;
    BasePoint.push_back(x);
    return 1;
}
int CreateLine(Line l) {
    if (BaseLine.count(l)) return BaseLine[l];
    int t = BaseLine.size();
    sourceLine[t+1] = l;
    return BaseLine[l] = t+1;
}
int CreateAngle(Angle a) {
    if (BaseAngle.count(a)) return BaseAngle[a];
    int t = BaseAngle.size();
    sourceAngle[t+1] = a;
    return BaseAngle[a] = t+1;
}
int CreateTriangle(Triangle a) {
    if (BaseTriangle.count(a)) return BaseTriangle[a];
    int t = BaseTriangle.size();
    sourceTriangle[t+1] = a;
    return BaseTriangle[a] = t+1;
}
bool isEqual(Angle a, int b) {
    return degree.count(a)&&degree[a]==b;
}
bool isEqual(Angle a, Angle b) {
    if (degree.count(a)&&degree.count(b)) return degree[a]==degree[b];
    if (degree2.count(a)&&degree2.count(b)) return degree2[a]==degree2[b];
```

```
    int u = Find(Angle_fa, CreateAngle(a)), v = Find(Angle_fa,
CreateAngle(b));
    return u==v;
}
bool isEqual(Line a, Line b) {
    if (length.count(a)&&length.count(b)) return length[a]==length[b];
    if (length2.count(a)&&length2.count(b)) return length2[a]==length2[b];
    int u = Find(Line_fa, CreateLine(a)), v = Find(Line_fa, CreateLine(b));
    return u==v;
}
void getValue(int x) {
    if (Value!=1e8) return;
    Value = x;
    for (auto &t:degree2) {
        auto v = t.second;
        degree[t.first] = Value*v.x+v.y;
    }
    degree2.clear();
    for (auto &t:length2) {
        auto v = t.second;
        length[t.first] = Value*v.x+v.y;
    }
    length2.clear();
}
//a=b
void getValue(Expression a, Expression b) {
    if (Value!=1e8) return;
    if (a==b) return;
    getValue((b.y-a.y)/(a.x-b.x));
}
//a+b=c
void getValue(Expression a, Expression b, Expression c) {
    if (Value!=1e8) return;
    getValue(a+b,c);
}
//ax+b=0
void getValue(int a, int b) {
    if (a==0||Value!=1e8) return;
    int x = -b/a;
    if (x>=0&&a*x+b==0) return getValue(x);
}
//ax^2+bx+c=0
void getValue(int a, int b, int c) {
    if (Value!=1e8) return;
    if (a==0) return getValue(b,c);
    double delta = sqrt(b*b-4*a*c);
    int x1 = (-b+delta)/(2*a), x2 = (-b-delta)/(2*a);
    if (x1>=0&&(long long)a*x1*x1+(long long)b*x1+c==0) return
getValue(x1);
    if (x2>=0&&(long long)a*x2*x2+(long long)b*x2+c==0) return
```

```
getValue(x2);
}
//a=b
void LineEquation(Line a, int b) {
    a = sourceLine[Find(Line_fa,CreateLine(a))];
    if (length2.count(a)) getValue(length2[a],Expression(0,b));
    length[a] = b;
}
//a=b
void LineEquation(Line a, Expression b) {
    if (b.x==0) return LineEquation(a,b.y);
    if (Value!=1e8) return LineEquation(a,b.y+b.x*Value);
    a = sourceLine[Find(Line_fa,CreateLine(a))];
    if (length2.count(a)) return getValue(length2[a],b);
    length2[a] = b;
}
//a=b
void LineEquation(Line a, Line b) {
    if (isEqual(a,b)) return;
    int u = Find(Line_fa,CreateLine(a)), v = Find(Line_fa,CreateLine(b));
    a = sourceLine[u], b = sourceLine[v];
    if (length.count(a)) {
        Line_fa[v] = u;
        return;
    }
    if (length.count(b)) {
        Line_fa[u] = v;
        return;
    }
    if (length2.count(a)&&length2.count(b)) {
        Line_fa[u] = v;
        return getValue(length2[a],length2[b]);
    }
    if (length2.count(a)) {
        Line_fa[v] = u;
        return;
    }
    if (length2.count(b)) {
        Line_fa[u] = v;
        return;
    }
    Line_fa[u] = v;
}
//a+b=c
map<pair<Line,Line>,Line> Base_Line_Equation2;
//a-b=c
map<pair<Line,Line>,Line> Base_Line_Equation3;
void LineEquation(Line a, Line b, Line c) {
    if (b<a) swap(a,b);
    if (Base_Line_Equation2.count({a,b})) {
        LineEquation(Base_Line_Equation2[{a,b}],c);
```

```
    }
    else Base_Line_Equation2[{a,b}]=c;
    if (Base_Line_Equation3.count({c,a})) {
        LineEquation(Base_Line_Equation3[{c,a}],b);
    }
    else Base_Line_Equation3[{c,a}]=b;
    if (Base_Line_Equation3.count({c,b})) {
        LineEquation(Base_Line_Equation3[{c,b}],a);
    }
    else Base_Line_Equation3[{c,b}]=a;
}
//a/b=c/d
map<pair<pair<Line,Line>,Line>,Line> Base_Line_Equation4;
void LineEquation(Line a, Line b, Line c, Line d) {
    if (isEqual(a,c)) return LineEquation(b,d);
    if (isEqual(b,d)) return LineEquation(a,c);
    if (isEqual(a,b)) return LineEquation(c,d);
    if (isEqual(c,d)) return LineEquation(a,b);
    pair<pair<Line,Line>,Line> ret = {{a,b},c};
    if (Base_Line_Equation4.count(ret)) {
        LineEquation(Base_Line_Equation4[ret],d);
    }
    else Base_Line_Equation4[ret]=d;
}
void AngleEquation(Angle a, int b) {
//  printf("[%c%c%c]=%d\n",a.x.x,a.y.x,a.z.x,b);
    a = sourceAngle[Find(Angle_fa,CreateAngle(a))];
    if (degree2.count(a)) getValue(degree2[a],Expression(0,b));
    degree[a] = b;
}
void AngleEquation(Angle a, Expression b) {
    if (b.x==0) return AngleEquation(a,b.y);
    if (Value!=1e8) return AngleEquation(a,b.y+b.x*Value);
    a = sourceAngle[Find(Angle_fa,CreateAngle(a))];
    if (degree2.count(a)) return getValue(degree2[a],b);
    degree2[a] = b;
}
void AngleEquation(Angle a, Angle b) {
//  printf("[%c%c%c]=[%c%c%c]\n",a.x.x,a.y.x,a.z.x,b.x.x,b.y.x,b.z.x);
    if (isEqual(a,b)) return;
    int u = Find(Angle_fa,CreateAngle(a)), v =
Find(Angle_fa,CreateAngle(b));
    a = sourceAngle[u], b = sourceAngle[v];
    if (degree.count(a)) {
        Angle_fa[v] = u;
        return;
    }
    if (degree.count(b)) {
        Angle_fa[u] = v;
        return;
```

```
    }
    if (degree2.count(a)&&degree2.count(b)) {
        Angle_fa[u] = v;
        return getValue(degree2[a],degree2[b]);
    }
    if (degree2.count(a)) {
        Angle_fa[v] = u;
        return;
    }
    if (degree2.count(b)) {
        Angle_fa[u] = v;
        return;
    }
    Angle_fa[u] = v;
}
//a+b=180
map<Angle,Angle> Base_Angle_Equation2;
void AngleEquation2(Angle a, Angle b) {
//  printf("[%c%c%c]+[%c%c%c]=180\n",a.x.x,a.y.x,a.z.x,b.x.x,b.y.x,b.z.x);
    if (isEqual(a,b)) return AngleEquation(a,90),AngleEquation(b,90);
    if (Base_Angle_Equation2.count(a)) {
        AngleEquation(Base_Angle_Equation2[a],b);
    }
    else Base_Angle_Equation2[a]=b;
    if (Base_Angle_Equation2.count(b)) {
        AngleEquation(Base_Angle_Equation2[b],a);
    }
    else Base_Angle_Equation2[b]=a;
}
//a+b=c
map<pair<Angle,Angle>,Angle> Base_Angle_Equation3;
//a-b=c
map<pair<Angle,Angle>,Angle> Base_Angle_Equation4;
void AngleEquation(Angle a, Angle b, Angle c) {
//
printf("[%c%c%c]+[%c%c%c]=[%c%c%c]\n",a.x.x,a.y.x,a.z.x,b.x.x,b.y.x,b.z.x,c
.x.x,c.y.x,c.z.x);
    if (b<a) swap(a,b);
    if (Base_Angle_Equation3.count({a,b})) {
        AngleEquation(Base_Angle_Equation3[{a,b}],c);
    }
    else Base_Angle_Equation3[{a,b}]=c;
    if (Base_Angle_Equation4.count({c,a})) {
        AngleEquation(Base_Angle_Equation4[{c,a}],b);
    }
    else Base_Angle_Equation3[{c,a}]=b;
    if (Base_Angle_Equation4.count({c,b})) {
        AngleEquation(Base_Angle_Equation4[{c,b}],a);
    }
    else Base_Angle_Equation4[{c,b}]=a;
}
```

```
void AngleEquation2(Angle a, Angle b, Angle c) {
//
printf("[%c%c%c]+[%c%c%c]+[%c%c%c]=180\n",a.x.x,a.y.x,a.z.x,b.x.x,b.y.x,b.z
.x,c.x.x,c.y.x,c.z.x);
    Angle_Equation e;
    e.var1 = CreateAngle(a);
    e.var2 = CreateAngle(b);
    e.var3 = CreateAngle(c);
    Base_Angle_Equation[3].push_back(e);
}
void CreateEquation(Term a, Term b) {
    if (a.type==1&&b.type==1) LineEquation(a.l,b.l);
    else if (a.type==1&&b.type==3) LineEquation(a.l,b.val);
    else if (a.type==3&&b.type==1) LineEquation(b.l,a.val);
    else if (a.type==3&&b.type==3) getValue(a.val,b.val);
    else if (a.type==2&&b.type==2) AngleEquation(a.ang,b.ang);
    else if (a.type==2&&b.type==3) AngleEquation(a.ang,b.val);
    else if (a.type==3&&b.type==2) AngleEquation(b.ang,a.val);
//    else throw;
}
bool isExist(Angle a) {return BaseAngle.count(a);}
bool isExist(Line l) {return BaseLine.count(l);}
bool isExist(Line l, char a) {
    if (!BaseLine.count(l)) return 0;
    if (a==l.x||a==l.y) return 1;
    if (!Point_Line.count(l)) return 0;
    for (auto &t:Point_Line[l]) if (t==a) return 1;
    return 0;
}
bool isProportional(Line a, Line b, Line c, Line d) {
    return 0;
}
bool isCollinear(char a, char b, char c) {
    if (a==b||a==c||b==c) return 1;
    Line AB(a,b), BC(b,c), CA(c,a);
    if (isExist(AB)&&isExist(AB,c)) return 1;
    if (isExist(BC)&&isExist(BC,a)) return 1;
    if (isExist(CA)&&isExist(CA,b)) return 1;
    return 0;
}
bool isParallel(Line a, Line b) {
    if (parallel.count({a,b})||parallel.count({b,a})) return 1;
    return 0;
}
bool isIntersect(Line a, Line b) {
    for (auto &t:BasePoint) if (isExist(a,t)&&isExist(b,t)) return 1;
    return 0;
}
char getNode(Line a, Line b) {
    for (auto &t:BasePoint) if
```

```
(isCollinear(a.x,a.y,t)&&isCollinear(b.x,b.y,t)) return t;
    return 0;
}
void PointLiesOnLine(Line a, char b) {
    LineEquation(Line(a.x,b),Line(a.y,b),a);
    int cnt = BasePoint.size();
    REP(i,0,cnt-1) {
        Angle x(a.x,b,BasePoint[i]);
        Angle y(a.y,b,BasePoint[i]);
        if (isExist(x)&&isExist(y)) AngleEquation2(x,y);
    }
}
void ThePythagoreanTheorem(Line a, Line b, Line c) {
    Line_Equation e;
    e.var1 = CreateLine(a);
    e.var2 = CreateLine(b);
    e.var3 = CreateLine(c);
    Base_Line_Equation[2].push_back(e);
}
void TrianglesTheorem(Triangle a, Triangle b) {
    Angle A1 = Angle(a.y,a.x,a.z);
    Angle A2 = Angle(b.y,b.x,b.z);
    Angle B1 = Angle(a.x,a.y,a.z);
    Angle B2 = Angle(b.x,b.y,b.z);
    Angle C1 = Angle(a.x,a.z,a.y);
    Angle C2 = Angle(b.x,b.z,b.y);
    Line AB1(a.x,a.y), BC1(a.y,a.z), CA1(a.z,a.x);
    Line AB2(b.x,b.y), BC2(b.y,b.z), CA2(b.z,b.x);
    //SSS
    if (isEqual(AB1,AB2)&&isEqual(BC1,BC2)&&isEqual(CA1,CA2)) {
        AngleEquation(A1,A2);
        AngleEquation(B1,B2);
        AngleEquation(C1,C2);
        return;
    }
    //SAS
    if (isEqual(AB1,AB2)&&isEqual(BC1,BC2)&&isEqual(B1,B2)) {
        LineEquation(CA1,CA2);
        AngleEquation(A1,A2);
        AngleEquation(C1,C2);
        return;
    }
    if (isEqual(AB1,AB2)&&isEqual(CA1,CA2)&&isEqual(A1,A2)) {
        LineEquation(BC1,BC2);
        AngleEquation(B1,B2);
        AngleEquation(C1,C2);
        return;
    }
    if (isEqual(BC1,BC2)&&isEqual(CA1,CA2)&&isEqual(C1,C2)) {
        LineEquation(AB1,AB2);
        AngleEquation(A1,A2);
```

```
        AngleEquation(B1,B2);
        return;
    }
    //ASA
    if (isEqual(A1,A2)&&isEqual(B1,B2)&&isEqual(AB1,AB2)) {
        LineEquation(CA1,CA2);
        LineEquation(BC1,BC2);
        AngleEquation(C1,C2);
        return;
    }
    if (isEqual(A1,A2)&&isEqual(C1,C2)&&isEqual(CA1,CA2)) {
        LineEquation(BC1,BC2);
        LineEquation(AB1,AB2);
        AngleEquation(B1,B2);
        return;
    }
    if (isEqual(B1,B2)&&isEqual(C1,C2)&&isEqual(BC1,BC2)) {
        LineEquation(AB1,AB2);
        LineEquation(CA1,CA2);
        AngleEquation(A1,A2);
        return;
    }
    //AAS
    if (isEqual(A1,A2)&&isEqual(B1,B2)) {
        if (isEqual(BC1,BC2)) {
            LineEquation(CA1,CA2);
            LineEquation(AB1,AB2);
            AngleEquation(C1,C2);
            return;
        }
        if (isEqual(CA1,CA2)) {
            LineEquation(AB1,AB2);
            LineEquation(BC1,BC2);
            AngleEquation(C1,C2);
            return;
        }
    }
    if (isEqual(A1,A2)&&isEqual(C1,C2)) {
        if (isEqual(BC1,BC2)) {
            LineEquation(CA1,CA2);
            LineEquation(AB1,AB2);
            AngleEquation(B1,B2);
            return;
        }
        if (isEqual(AB1,AB2)) {
            LineEquation(CA1,CA2);
            LineEquation(BC1,BC2);
            AngleEquation(B1,B2);
            return;
        }
```

```
    }
    if (isEqual(B1,B2)&&isEqual(C1,C2)) {
        if (isEqual(CA1,CA2)) {
            LineEquation(AB1,AB2);
            LineEquation(BC1,BC2);
            AngleEquation(A1,A2);
            return;
        }
        if (isEqual(AB1,AB2)) {
            LineEquation(BC1,BC2);
            LineEquation(CA1,CA2);
            AngleEquation(A1,A2);
            return;
        }
    }
    //HL
    if (isEqual(A1,90)&&isEqual(A2,90)&&isEqual(BC1,BC2)) {
        if (isEqual(AB1,AB2)) {
            LineEquation(CA1,CA2);
            AngleEquation(B1,B2);
            AngleEquation(C1,C2);
            return;
        }
        if (isEqual(CA1,CA2)) {
            LineEquation(AB1,AB2);
            AngleEquation(B1,B2);
            AngleEquation(C1,C2);
            return;
        }
    }
    if (isEqual(B1,90)&&isEqual(B2,90)&&isEqual(CA1,CA2)) {
        if (isEqual(AB1,AB2)) {
            LineEquation(BC1,BC2);
            AngleEquation(A1,A2);
            AngleEquation(C1,C2);
            return;
        }
        if (isEqual(BC1,BC2)) {
            LineEquation(AB1,AB2);
            AngleEquation(A1,A2);
            AngleEquation(C1,C2);
            return;
        }
    }
    if (isEqual(C1,90)&&isEqual(C2,90)&&isEqual(AB1,AB2)) {
        if (isEqual(CA1,CA2)) {
            LineEquation(BC1,BC2);
            AngleEquation(A1,A2);
            AngleEquation(B1,B2);
            return;
        }
```

```
        if (isEqual(BC1,BC2)) {
            LineEquation(CA1,CA2);
            AngleEquation(A1,A2);
            AngleEquation(B1,B2);
            return;
        }
    }
    //AA
    if (isEqual(A1,A2)&&isEqual(B1,B2)) {
        AngleEquation(C1,C2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(AB1,AB2,CA1,CA2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    if (isEqual(A1,A2)&&isEqual(C1,C2)) {
        AngleEquation(B1,B2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(AB1,AB2,CA1,CA2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    if (isEqual(B1,B2)&&isEqual(C1,C2)) {
        AngleEquation(A1,A2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(AB1,AB2,CA1,CA2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    //SAS
    if (isProportional(AB1,AB2,BC1,BC2)&&isEqual(B1,B2)) {
        AngleEquation(A1,A2);
        AngleEquation(C1,C2);
        LineEquation(AB1,AB2,CA1,CA2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    if (isProportional(AB1,AB2,CA1,CA2)&&isEqual(A1,A2)) {
        AngleEquation(B1,B2);
        AngleEquation(C1,C2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    if (isProportional(BC1,BC2,CA1,CA2)&&isEqual(C1,C2)) {
        AngleEquation(A1,A2);
        AngleEquation(B1,B2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(AB1,AB2,CA1,CA2);
        return;
```

```
    }
    //SSS
    if (isProportional(AB1,AB2,BC1,BC2)&&isProportional(AB1,AB2,CA1,CA2)) {
        AngleEquation(A1,A2);
        AngleEquation(B1,B2);
        AngleEquation(C1,C2);
        return;
    }
    //平行
    if (isParallel(AB1,AB2)&&isParallel(BC1,BC2)&&isParallel(CA1,CA2)) {
        AngleEquation(A1,A2);
        AngleEquation(B1,B2);
        AngleEquation(C1,C2);
        LineEquation(AB1,AB2,BC1,BC2);
        LineEquation(AB1,AB2,CA1,CA2);
        LineEquation(BC1,BC2,CA1,CA2);
        return;
    }
    //HL
    if (isEqual(A1,90)&&isEqual(A2,90)) {
        if (isProportional(BC1,BC2,AB1,AB2)) {
            AngleEquation(B1,B2);
            AngleEquation(C1,C2);
            LineEquation(AB1,AB2,CA1,CA2);
            LineEquation(BC1,BC2,CA1,CA2);
        }
        if (isProportional(BC1,BC2,CA1,CA2)) {
            LineEquation(AB1,AB2,BC1,BC2);
            LineEquation(AB1,AB2,CA1,CA2);
        }
    }
    if (isEqual(B1,90)&&isEqual(B2,90)) {
        if (isProportional(CA1,CA2,AB1,AB2)) {
            AngleEquation(A1,A2);
            AngleEquation(C1,C2);
            LineEquation(AB1,AB2,CA1,CA2);
            LineEquation(BC1,BC2,CA1,CA2);
        }
        if (isProportional(CA1,CA2,BC1,BC2)) {
            AngleEquation(A1,A2);
            AngleEquation(C1,C2);
            LineEquation(BC1,BC2,CA1,CA2);
            LineEquation(AB1,AB2,BC1,BC2);
        }
    }
    if (isEqual(C1,90)&&isEqual(C2,90)) {
        if (isProportional(AB1,AB2,CA1,CA2)) {
            AngleEquation(B1,B2);
            AngleEquation(C1,C2);
            LineEquation(BC1,BC2,CA1,CA2);
            LineEquation(AB1,AB2,BC1,BC2);
```

```
        }
        if (isProportional(AB1,AB2,BC1,BC2)) {
            AngleEquation(B1,B2);
            AngleEquation(C1,C2);
            LineEquation(BC1,BC2,CA1,CA2);
            LineEquation(AB1,AB2,CA1,CA2);
        }
    }
}
Logic_Form dfs(string &s, int l, int r) {
    Logic_Form log;
    if (s[l]==' ') return dfs(s,l+1,r);
    if (s.substr(l,5)=="Line(") {
        char A=0, B=0;
        REP(i,l+5,r) if (isupper(s[i])) {
            if (A==0) A = s[i];
            else B = s[i];
        }
        log.l = Line(A,B);
        CreateLine(log.l);
        return log;
    }
    if (s.substr(l,6)=="Angle(") {
        char A=0,B=0,C=0;
        REP(i,l+6,r) if (isupper(s[i])) {
            if (A==0) A = s[i];
            else if (B==0) B = s[i];
            else C = s[i];
        }
        CreateLine(Line(A,B));
        CreateLine(Line(B,C));
        log.ang = Angle(A,B,C);
        return log;
    }
    if (s.substr(l,7)=="Circle(") {
        char A=0;
        REP(i,l+7,r) if (isupper(s[i])) A = s[i];
        log.c = A;
        return log;
    }
    if (s.substr(l,9)=="LengthOf(") {
        log.t.type = 1;
        log.t.l = dfs(s,l+9,r).l;
        return log;
    }
    if (s.substr(l,10)=="MeasureOf(") {
        log.t.type = 2;
        log.t.ang = dfs(s,l+10,r).ang;
        return log;
    }
```

```
    if (s.substr(l,7)=="Equals(") {
        Term lhs, rhs;
        int sum = 0;
        REP(i,l+7,r) {
            if (s[i]=='(') ++sum;
            if (s[i]==')') --sum;
            if (s[i]==','&&!sum) {
                lhs = dfs(s,l+7,i-1).t;
                rhs = dfs(s,i+1,r-1).t;
                break;
            }
        }
        CreateEquation(lhs,rhs);
        return log;
    }
    if (s.substr(l,14)=="Perpendicular(") {
        Line lhs, rhs;
        int sum = 0;
        REP(i,l+14,r) {
            if (s[i]=='(') ++sum;
            if (s[i]==')') --sum;
            if (s[i]==','&&!sum) {
                lhs = dfs(s,l+14,i-1).l;
                rhs = dfs(s,i+1,r-1).l;
                break;
            }
        }
        perpendicular[{lhs,rhs}] = 1;
        return log;
    }
    if (s.substr(l,9)=="Parallel(") {
        Line lhs, rhs;
        int sum = 0;
        REP(i,l+9,r) {
            if (s[i]=='(') ++sum;
            if (s[i]==')') --sum;
            if (s[i]==','&&!sum) {
                lhs = dfs(s,l+9,i-1).l;
                rhs = dfs(s,i+1,r-1).l;
                break;
            }
        }
        parallel[{lhs,rhs}] = 1;
        return log;
    }
    if (s.substr(l,16)=="PointLiesOnLine(") {
        char lhs=0;
        Line rhs;
        int sum = 0;
        REP(i,l+16,r) {
            if (s[i]=='(') ++sum;
```

```
            if (s[i]==')') --sum;
            if (s[i]==','&&!sum) {
                REP(j,l+16,i-1) if (isupper(s[j])) {
                    lhs = s[j];
                    break;
                }
                rhs = dfs(s,i+1,r-1).l;
                break;
            }
        }
        Point_Line[rhs].push_back(lhs);
        return log;
    }
    if (s.substr(l,18)=="PointLiesOnCircle(") {
        char lhs=0;
        char rhs=0;
        int sum = 0;
        REP(i,l+18,r) {
            if (s[i]=='(') ++sum;
            if (s[i]==')') --sum;
            if (s[i]==','&&!sum) {
                REP(j,l+18,i-1) if (isupper(s[j])) {
                    lhs = s[j];
                    break;
                }
                rhs = dfs(s,i+1,r-1).c;
                break;
            }
        }
        Point_Circle[rhs].push_back(lhs);
        return log;
    }
    if (s.substr(l,5)=="Find(") {
        Question = dfs(s,l+5,r-1).t;
        return log;
    }
    log.t.type = 3;
    int pos = 0, num = 0;
    REP(i,l,r) {
        if (s[i]=='x') pos = i;
        if (isdigit(s[i])) num = num*10+s[i]-'0';
    }
    if (!pos) {
        log.t.val = Expression(0,num);
        return log;
    }
    int sign = 0, sign_pos = 0;
    REP(i,l,r) {
        if (s[i]=='+') sign = 1, sign_pos = i;
        if (s[i]=='-') sign = -1, sign_pos = i;
```

```
    }
    if (!sign) {
        int num = 0, x = 1;
        REP(i,l,pos-1) {
            if (isdigit(s[i])) num=num*10+s[i]-'0';
        }
        if (num) x *= num;
        num = 0;
        REP(i,pos+1,r) {
            if (isdigit(s[i])) num=num*10+s[i]-'0';
        }
        if (num) x *= num;
        if (Value==1e8) log.t.val = Expression(x,0);
        else log.t.val = Expression(0,Value*x);
        return log;
    }
    if (sign_pos<pos) {
        int num = 0;
        REP(i,l,sign_pos-1) {
            if (isdigit(s[i])) num=num*10+s[i]-'0';
        }
        int x = 1, tmp = 0;
        REP(i,sign_pos+1,pos-1) {
            if (isdigit(s[i])) tmp=tmp*10+s[i]-'0';
        }
        if (tmp) x *= tmp;
        tmp = 0;
        REP(i,pos+1,r) {
            if (isdigit(s[i])) tmp=tmp*10+s[i]-'0';
        }
        if (tmp) x *= tmp;
        if (Value==1e8) log.t.val = Expression(sign*x,num);
        else log.t.val = Expression(0,num+sign*x*Value);
        return log;
    }
    if (sign_pos>pos) {
        int num = 0;
        REP(i,sign_pos+1,r) {
            if (isdigit(s[i])) num=num*10+s[i]-'0';
        }
        int x = 1, tmp = 0;
        REP(i,l,pos-1) {
            if (isdigit(s[i])) tmp=tmp*10+s[i]-'0';
        }
        if (tmp) x *= tmp;
        tmp = 0;
        REP(i,pos+1,sign_pos-1) {
            if (isdigit(s[i])) tmp=tmp*10+s[i]-'0';
        }
        if (tmp) x *= tmp;
        if (Value==1e8) log.t.val = Expression(x,sign*num);
```

```
        else log.t.val = Expression(0,sign*num+x*Value);
        return log;
    }
//    throw;
}
//a/b=c/d
void solveProportional(Line a, Line b, Line c, Line d) {
    a = sourceLine[Find(Line_fa,CreateLine(a))];
    b = sourceLine[Find(Line_fa,CreateLine(b))];
    c = sourceLine[Find(Line_fa,CreateLine(c))];
    d = sourceLine[Find(Line_fa,CreateLine(d))];
    Expression
va=length.count(a)?Expression(0,length[a]):length2.count(a)?length2[a]:Expr
ession(1e8,1e8);
    Expression
vb=length.count(b)?Expression(0,length[b]):length2.count(b)?length2[b]:Expr
ession(1e8,1e8);
    Expression
vc=length.count(c)?Expression(0,length[c]):length2.count(c)?length2[c]:Expr
ession(1e8,1e8);
    Expression
vd=length.count(d)?Expression(0,length[d]):length2.count(d)?length2[d]:Expr
ession(1e8,1e8);
    if ((va+vb+vc+vd).y>=1.5e8) return;
    if (va.y==1e8) {
        //a=cb/d
        if (vd.x==0&&(long long)vc.y*vb.y%vd.y==0) {
            if (vc.x==0&&vb.x==0) LineEquation(a,(long
long)vc.y*vb.y/vd.y);
            else if (vc.x==0) {
                if ((long long)vc.y*vb.x%vd.y==0) {
                    LineEquation(a,Expression((long
long)vc.y*vb.x/vd.y,(long long)vc.y*vb.y/vd.y));
                }
            }
            else if (vb.x==0) {
                if ((long long)vb.y*vc.x%vd.y==0) {
                    LineEquation(a,Expression((long
long)vb.y*vc.x/vd.y,(long long)vc.y*vb.y/vd.y));
                }
            }
        }
        return;
    }
    if (vb.y==1e8) {
        //b=ad/c
        if (vc.x==0&&(long long)va.y*vd.y%vc.y==0) {
            if (va.x==0&&vd.x==0) LineEquation(b,(long
long)va.y*vd.y/vc.y);
            else if (va.x==0) {
```

```
                if ((long long)va.y*vd.x%vc.y==0) {
                    LineEquation(b,Expression((long
long)va.y*vd.x/vc.y,(long long)va.y*vd.y/vc.y));
                }
            }
            else if (vd.x==0) {
                if ((long long)vd.y*va.x%vc.y==0) {
                    LineEquation(b,Expression((long
long)vd.y*va.x/vc.y,(long long)va.y*vd.y/vc.y));
                }
            }
        }
        return;
    }
    if (vc.y==1e8) {
        //c=ad/b
        if (vb.x==0&&(long long)va.y*vd.y%vb.y==0) {
            if (va.x==0&&vd.x==0) LineEquation(c,(long
long)va.y*vd.y/vb.y);
            else if (va.x==0) {
                if ((long long)va.y*vd.x%vb.y==0) {
                    LineEquation(c,Expression((long
long)va.y*vd.x/vb.y,(long long)va.y*vd.y/vb.y));
                }
            }
            else if (vd.x==0) {
                if ((long long)vd.y*va.x%vb.y==0) {
                    LineEquation(c,Expression((long
long)vd.y*va.x/vb.y,(long long)va.y*vd.y/vb.y));
                }
            }
        }
        return;
    }
    if (vd.y==1e8) {
        //d=cb/a
        if (va.x==0&&(long long)vc.y*vb.y%va.y==0) {
            if (vc.x==0&&vb.x==0) LineEquation(d,(long
long)vc.y*vb.y/va.y);
            else if (vc.x==0) {
                if ((long long)vc.y*vb.x%va.y==0) {
                    LineEquation(d,Expression((long
long)vc.y*vb.x/va.y,(long long)vc.y*vb.y/va.y));
                }
            }
            else if (vb.x==0) {
                if ((long long)vb.y*vc.x%va.y==0) {
                    LineEquation(d,Expression((long
long)vb.y*vc.x/va.y,(long long)vc.y*vb.y/va.y));
                }
            }
```

```
        }
        return;
    }
    //a.xd.xX^2+a.x*d.yX+a.y*d.xX+a.y*d.y=c.xb.xX^2+c.yb.xX+c.xb.yX+c.yb.y
    //(a.xd.x-c.xb.x) X^2 + (a.x*d.y+a.y*d.x-c.yb.x-c.xb.y) X + a.yd.y-
c.yb.y = 0
    getValue(va.x*vd.x-vc.x*vb.x,va.x*vd.y+va.y*vd.x-vc.y*vb.x-
vc.x*vb.y,va.y*vd.y-vc.y*vb.y);
}
void solveAngle() {
    int tot = BaseAngle.size();
    REP(i,1,tot) {
        Angle u = sourceAngle[Find(Angle_fa,i)];
        if (degree.count(u)) degree[sourceAngle[i]]=degree[u];
        else if (degree2.count(u)) degree2[sourceAngle[i]]=degree2[u];
    }
    for (auto &e:Base_Angle_Equation2) {
        auto L = e.first, R = e.second;
        L = sourceAngle[Find(Angle_fa,CreateAngle(L))];
        R = sourceAngle[Find(Angle_fa,CreateAngle(R))];
        if (isEqual(L,R)) {
            AngleEquation(L,90);
            AngleEquation(R,90);
            continue;
        }
        if (degree.count(L)) {
            AngleEquation(R,180-degree[L]);
            continue;
        }
        if (degree.count(R)) {
            AngleEquation(L,180-degree[R]);
            continue;
        }
        if (degree2.count(L)&&degree2.count(R)) {
            getValue(degree2[L],degree2[R],Expression(0,180));
            continue;
        }
        if (degree2.count(L)) {
            AngleEquation(R,Expression(0,180)-degree2[L]);
            continue;
        }
        if (degree2.count(R)) {
            AngleEquation(L,Expression(0,180)-degree2[R]);
            continue;
        }
    }
    for (auto &e:Base_Angle_Equation3) {
        auto u = e.first.first, v = e.first.second, w = e.second;
        u = sourceAngle[Find(Angle_fa,CreateAngle(u))];
        v = sourceAngle[Find(Angle_fa,CreateAngle(v))];
```

```
        w = sourceAngle[Find(Angle_fa,CreateAngle(w))];
        if (degree.count(u)&&degree.count(v)) {
            AngleEquation(w,degree[u]+degree[v]);
            continue;
        }
        if (degree.count(u)&&degree.count(w)) {
            AngleEquation(v,degree[w]-degree[u]);
            continue;
        }
        if (degree.count(v)&&degree.count(w)) {
            AngleEquation(u,degree[w]-degree[v]);
            continue;
        }
        if (degree.count(u)) {
            if (degree2.count(w)&&degree2.count(v))
getValue(Expression(0,degree[u]),degree2[v],degree2[w]);
            else if (degree2.count(w)) AngleEquation(v,degree2[w]-
Expression(0,degree[u]));
            else if (degree2.count(v))
AngleEquation(w,degree2[v]+Expression(0,degree[u]));
            continue;
        }
        if (degree.count(v)) {
            if (degree2.count(w)&&degree2.count(u))
getValue(degree2[u],Expression(0,degree[v]),degree2[w]);
            else if (degree2.count(w)) AngleEquation(u,degree2[w]-
Expression(0,degree[v]));
            else if (degree2.count(u))
AngleEquation(w,degree2[u]+Expression(0,degree[v]));
            continue;
        }
        if (degree.count(w)) {
            if (degree2.count(u)&&degree2.count(v))
getValue(degree2[u],degree2[v],Expression(0,degree[w]));
            else if (degree2.count(u))
AngleEquation(v,Expression(0,degree[w])-degree2[u]);
            else if (degree2.count(v))
AngleEquation(u,Expression(0,degree[w])-degree2[v]);
            continue;
        }
        if (degree2.count(u)&&degree2.count(v)&&degree2.count(w)) {
            getValue(degree2[u],degree2[v],degree2[w]);
            continue;
        }
        if (degree2.count(u)&&degree2.count(v)) {
            AngleEquation(w,degree2[u]+degree2[v]);
            continue;
        }
        if (degree2.count(u)&&degree2.count(w)) {
            AngleEquation(v,degree2[w]-degree2[u]);
            continue;
```

```
        }
        if (degree2.count(v)&&degree2.count(w)) {
            AngleEquation(u,degree2[w]-degree2[v]);
            continue;
        }
    }
    for (auto &e:Base_Angle_Equation[3]) {
        auto u = sourceAngle[e.var1], v = sourceAngle[e.var2], w =
sourceAngle[e.var3];
        u = sourceAngle[Find(Angle_fa,CreateAngle(u))];
        v = sourceAngle[Find(Angle_fa,CreateAngle(v))];
        w = sourceAngle[Find(Angle_fa,CreateAngle(w))];
        if (isEqual(u,v)&&isEqual(u,w)) {
            AngleEquation(u,60);
            AngleEquation(v,60);
            AngleEquation(w,60);
            continue;
        }
        if (degree.count(u)&&degree.count(v)) {
            AngleEquation(w,180-degree[u]-degree[v]);
            continue;
        }
        if (degree.count(u)&&degree.count(w)) {
            AngleEquation(v,180-degree[u]-degree[w]);
            continue;
        }
        if (degree.count(w)&&degree.count(v)) {
            AngleEquation(u,180-degree[v]-degree[w]);
            continue;
        }
        if (degree.count(u)) {
            if (degree2.count(w)&&degree2.count(v))
getValue(degree2[v],degree2[w],Expression(0,180-degree[u]));
            else if (degree2.count(w)) AngleEquation(v,Expression(0,180-
degree[u])-degree2[w]);
            else if (degree2.count(v)) AngleEquation(w,Expression(0,180-
degree[u])-degree2[v]);
            continue;
        }
        if (degree.count(v)) {
            if (degree2.count(w)&&degree2.count(u))
getValue(degree2[u],degree2[w],Expression(0,180-degree[v]));
            else if (degree2.count(w)) AngleEquation(u,Expression(0,180-
degree[v])-degree2[w]);
            else if (degree2.count(u)) AngleEquation(w,Expression(0,180-
degree[u])-degree2[u]);
            continue;
        }
        if (degree.count(w)) {
            if (degree2.count(u)&&degree2.count(v))
```

```
getValue(degree2[u],degree2[v],Expression(0,180-degree[w]));
            else if (degree2.count(u)) AngleEquation(v,Expression(0,180-
degree[w])-degree2[u]);
            else if (degree2.count(v)) AngleEquation(u,Expression(0,180-
degree[w])-degree2[v]);
            continue;
        }
        if (degree2.count(u)&&degree2.count(v)&&degree2.count(w)) {
            getValue(degree2[u]+degree2[v],degree2[w],Expression(0,180));
            continue;
        }
        if (degree2.count(u)&&degree2.count(v)) {
            AngleEquation(w,Expression(0,180)-degree2[u]-degree2[v]);
            continue;
        }
        if (degree2.count(u)&&degree2.count(w)) {
            AngleEquation(v,Expression(0,180)-degree2[u]-degree2[w]);
            continue;
        }
        if (degree2.count(v)&&degree2.count(w)) {
            AngleEquation(u,Expression(0,180)-degree2[v]-degree2[w]);
            continue;
        }
    }
    REP(i,1,tot) {
        Angle u = sourceAngle[Find(Angle_fa,i)];
        if (degree.count(u)) degree[sourceAngle[i]]=degree[u];
        else if (degree2.count(u)) degree2[sourceAngle[i]]=degree2[u];
    }
}
void solveLine() {
    int tot = BaseLine.size();
    REP(i,1,tot) {
        Line u = sourceLine[Find(Line_fa,i)];
        if (length.count(u)) length[sourceLine[i]]=length[u];
        else if (length2.count(u)) length2[sourceLine[i]]=length2[u];
    }
    for (auto &e:Base_Line_Equation2) {
        auto u = e.first.first, v = e.first.second, w = e.second;
        u = sourceLine[Find(Line_fa,CreateLine(u))];
        v = sourceLine[Find(Line_fa,CreateLine(v))];
        w = sourceLine[Find(Line_fa,CreateLine(w))];
        if (length.count(u)&&length.count(v)) {
            LineEquation(w,length[u]+length[v]);
            continue;
        }
        if (length.count(u)&&length.count(w)) {
            LineEquation(v,length[w]-length[u]);
            continue;
        }
        if (length.count(v)&&length.count(w)) {
```

```
            LineEquation(u,length[w]-length[v]);
            continue;
        }
        if (length.count(u)) {
            if (length2.count(w)&&length2.count(v))
getValue(Expression(0,length[u]),length2[v],length2[w]);
            else if (length2.count(w)) LineEquation(v,length2[w]-
Expression(0,length[u]));
            else if (length2.count(v))
LineEquation(w,length2[v]+Expression(0,length[u]));
            continue;
        }
        if (length.count(v)) {
            if (length2.count(w)&&length2.count(u))
getValue(length2[u],Expression(0,length[v]),length2[w]);
            else if (length2.count(w)) LineEquation(u,length2[w]-
Expression(0,length[v]));
            else if (length2.count(u)) LineEquation(w,length2[u]-
Expression(0,length[v]));
            continue;
        }
        if (length.count(w)) {
            if (length2.count(u)&&length2.count(v))
getValue(length2[u],length2[v],Expression(0,length[w]));
            else if (length2.count(u))
LineEquation(v,Expression(0,length[w])-length2[u]);
            else if (length2.count(v))
LineEquation(u,Expression(0,length[w])-length2[v]);
            continue;
        }
        if (length2.count(u)&&length2.count(v)&&length2.count(w)) {
            getValue(length2[u],length2[v],length2[w]);
            continue;
        }
        if (length2.count(u)&&length2.count(v)) {
            LineEquation(w,length2[u]+length2[v]);
            continue;
        }
        if (length2.count(u)&&length2.count(w)) {
            LineEquation(v,length2[w]-length2[u]);
            continue;
        }
        if (length2.count(v)&&length2.count(w)) {
            LineEquation(u,length2[w]-length2[v]);
            continue;
        }
    }
    for (auto &e:Base_Line_Equation4) {
        auto
a=e.first.first.first,b=e.first.first.second,c=e.first.second,d=e.second;
```

```
        //a/b=c/d
        solveProportional(a,b,c,d);
        pair<Line,Line> u(a,b), v(c,d);
        if (u.second<u.first) swap(u.second,u.first);
        if (v.second<v.first) swap(v.second,v.first);
        if (Base_Line_Equation2.count(u)&&Base_Line_Equation2.count(v)) {
            //(a+b)/b=(c+d)/d
solveProportional(Base_Line_Equation2[u],b,Base_Line_Equation2[v],d);
            //(a+b)/a=(c+d)/c
solveProportional(Base_Line_Equation2[u],a,Base_Line_Equation2[v],c);
        }
        u = {a,b}, v = {c,d};
        if (Base_Line_Equation3.count(u)&&Base_Line_Equation3.count(v)) {
            //(a-b)/b=(c-d)/d
solveProportional(Base_Line_Equation3[u],b,Base_Line_Equation3[v],d);
            //(a-b)/a=(c-d)/c
solveProportional(Base_Line_Equation3[u],a,Base_Line_Equation3[v],c);
        }
        u = {b,a}, v = {d,c};
        if (Base_Line_Equation3.count(u)&&Base_Line_Equation3.count(v)) {
            //(b-a)/b=(d-c)/d
solveProportional(Base_Line_Equation3[u],b,Base_Line_Equation3[v],d);
            //(b-a)/a=(d-c)/c
solveProportional(Base_Line_Equation3[u],a,Base_Line_Equation3[v],c);
        }
    }
    for (auto &e:Base_Line_Equation[2]) {
        auto
a=sourceLine[e.var1],b=sourceLine[e.var2],c=sourceLine[e.var3];
        a = sourceLine[Find(Line_fa,CreateLine(a))];
        b = sourceLine[Find(Line_fa,CreateLine(b))];
        c = sourceLine[Find(Line_fa,CreateLine(c))];
        Expression
va=length.count(a)?Expression(0,length[a]):length2.count(a)?length2[a]:Expr
ession(1e8,1e8);
        Expression
vb=length.count(b)?Expression(0,length[b]):length2.count(b)?length2[b]:Expr
ession(1e8,1e8);
        Expression
vc=length.count(c)?Expression(0,length[c]):length2.count(c)?length2[c]:Expr
ession(1e8,1e8);
        if ((va+vb+vc).y>=1.5e8) continue;
        if (va.y==1e8) {
            if (vc.x*vc.x==vb.x*vb.x&&vc.x*vc.y==vb.x*vb.y) {
                int u = sqrt(vc.y*vc.y-vb.y*vb.y);
                if (u*u+vb.y*vb.y==vc.y*vc.y) LineEquation(a,u);
            }
            continue;
        }
        if (vb.y==1e8) {
            if (vc.x*vc.x==va.x*va.x&&vc.x*vc.y==va.x*va.y) {
```

```
                int u = sqrt(vc.y*vc.y-va.y*va.y);
                if (u*u+va.y*va.y==vc.y*vc.y) LineEquation(b,u);
            }
            continue;
        }
        if (vc.y==1e8) {
            //(a.xX+a.y)^2+(b.xX+b.y)^2
            if (va.x==0&&vb.x==0) {
                int u = sqrt(va.y*va.y+vb.y*vb.y);
                if (va.y*va.y+vb.y*vb.y==u*u) LineEquation(c,u);
            }
            continue;
        }
        //(a.xX+a.y)^2+(b.xX+b.y)^2=(c.xX+c.y)^2
        //(a.x^2+b.x^2-c.x^2) X^2 + (2a.x*a.y+2b.x*b.y-2c.x*c.y) X
+a.y^2+b.y^2-c.y^2 = 0
        getValue(va.x*va.x+vb.x*vb.x-
vc.x*vc.x,2*va.x*va.y+2*vb.x*vb.y-2*vc.x*vc.y,va.y*va.y+vb.y*vb.y-
vc.y*vc.y);
    }
    REP(i,1,tot) {
        Line u = sourceLine[Find(Line_fa,i)];
        if (length.count(u)) length[sourceLine[i]]=length[u];
        else if (length2.count(u)) length2[sourceLine[i]]=length2[u];
    }
}
void solve() {
    solveAngle();
    solveLine();
    int cnt = BaseTriangle.size();
    REP(i,1,cnt) {
        auto &t = sourceTriangle[i];
        //等角对等边
        Angle xyz(t.x,t.y,t.z);
        Angle yzx(t.y,t.z,t.x);
        Angle zxy(t.z,t.x,t.y);
        Line xy(t.x,t.y),yz(t.y,t.z),zx(t.z,t.x);
        if (isEqual(xyz,yzx)) LineEquation(xy,zx);
        if (isEqual(xy,zx)) AngleEquation(xyz,yzx);
        if (isEqual(xyz,zxy)) LineEquation(zx,yz);
        if (isEqual(zx,yz)) AngleEquation(xyz,zxy);
        if (isEqual(zxy,yzx)) LineEquation(xy,yz);
        if (isEqual(xy,yz)) AngleEquation(zxy,yzx);
        //勾股定理
        if (isEqual(xyz,90)) ThePythagoreanTheorem(xy,yz,zx);
        if (isEqual(yzx,90)) ThePythagoreanTheorem(zx,yz,xy);
        if (isEqual(zxy,90)) ThePythagoreanTheorem(xy,zx,yz);
        REP(j,i+1,cnt) {
            auto &tt = sourceTriangle[j];
            //相似和全等
```

```
            TrianglesTheorem(t,tt);
        }
    }
}
int check() {
    if (Question.type==1) {
        if (length.count(Question.l)) {
            printf("%d\n", length[Question.l]);
            return 1;
        }
    }
    if (Question.type==2) {
        if (degree.count(Question.ang)) {
            printf("%d\n", degree[Question.ang]);
            return 1;
        }
    }
    if (Question.type==3) {
        if (Value!=1e8) {
            printf("%d\n", Value*Question.val.x+Question.val.y);
            return 1;
        }
    }
    return 0;
}

void work() {
    REP(i,1,5555-1) Line_fa[i]=Angle_fa[i]=i;
    BasePoint.clear();
    BaseLine.clear();
    length.clear();
    length2.clear();
    parallel.clear();
    perpendicular.clear();
    Point_Line.clear();
    degree.clear();
    degree2.clear();
    BaseAngle.clear();
    BaseTriangle.clear();
    Point_Circle.clear();
    for (auto &t:Base_Line_Equation) t.clear();
    for (auto &t:Base_Angle_Equation) t.clear();
    Value = 1e8;
    Base_Line_Equation2.clear();
    Base_Line_Equation3.clear();
    Base_Line_Equation4.clear();
    Base_Angle_Equation2.clear();
    Base_Angle_Equation3.clear();
    Base_Angle_Equation4.clear();
    string s;
    int n;
```

```
    cin>>n;
    cin.ignore();
    getline(cin,s);
    stringstream ss;
    ss<<s;
    while (ss>>s) CreatePoint(s[0]);
    getline(cin,s);
    stringstream ss2;
    ss2<<s;
    while (ss2>>s) CreateLine(Line(s[0],s[1]));
    REP(i,1,n) {
        getline(cin,s);
        dfs(s,0,s.size()-1);
    }
    int cnt = BasePoint.size();
    //构造角
    REP(i,0,cnt-1) REP(j,0,cnt-1) REP(k,0,cnt-1) if (i!=j&&i!=k&&j!=k) {
        if (isCollinear(BasePoint[i],BasePoint[j],BasePoint[k])) continue;
        Line AB(BasePoint[i],BasePoint[j]);
        Line AC(BasePoint[i],BasePoint[k]);
        Line BC(BasePoint[j],BasePoint[j]);
        if (isExist(AB)&&isExist(AC)&&!isExist(BC,BasePoint[i])) {
            CreateAngle(Angle(BasePoint[j],BasePoint[i],BasePoint[k]));
        }
    }
    cnt = BaseAngle.size();
    REP(i,1,cnt) REP(j,i+1,cnt) {
        Angle a = sourceAngle[i];
        Angle b = sourceAngle[j];
        if (a.y==b.y) {
            //重合的角
            if
((isExist(Line(a.x,a.y),b.x)||isExist(Line(b.x,a.y),a.x))&&(isExist(Line(a.
z,a.y),b.z)||isExist(Line(b.z,a.y),a.z))) {
                AngleEquation(a,b);
            }
            else if
((isExist(Line(a.x,a.y),b.z)||isExist(Line(b.z,a.y),a.x))&&(isExist(Line(a.
z,a.y),b.x)||isExist(Line(b.x,a.y),a.z))) {
                AngleEquation(a,b);
            }
        }
    }
    cnt = BasePoint.size();
    REP(i,0,cnt-1) REP(j,0,cnt-1) if (i!=j) {
        REP(x,0,cnt-1) REP(y,0,cnt-1) if (i!=x&&i!=y&&j!=x&&j!=y&&x!=y) {
            Angle a(BasePoint[j],BasePoint[i],BasePoint[x]);
            Angle b(BasePoint[j],BasePoint[i],BasePoint[y]);
            Angle c(BasePoint[x],BasePoint[i],BasePoint[y]);
            //if (isExist(a)&&isExist(b)&&isExist(c)) AngleEquation(a,b,c);
```

```
        }
    }
    //构造三角形
    REP(i,0,cnt-1) REP(j,0,cnt-1) REP(k,0,cnt-1) if (i!=j&&i!=k&&j!=k) {
        if (isCollinear(BasePoint[i],BasePoint[j],BasePoint[k])) continue;
        Line AB(BasePoint[i],BasePoint[j]);
        Line BC(BasePoint[j],BasePoint[k]);
        Line CA(BasePoint[k],BasePoint[i]);
        if (isExist(AB)&&isExist(BC)&&isExist(CA)) {
CreateTriangle(Triangle(BasePoint[i],BasePoint[j],BasePoint[k]));
        }
    }
    cnt = BaseTriangle.size();
    REP(i,1,cnt) {
        Triangle t = sourceTriangle[i];
        Angle xyz(t.x,t.y,t.z);
        Angle yzx(t.y,t.z,t.x);
        Angle zxy(t.z,t.x,t.y);
        AngleEquation2(xyz,yzx,zxy);
    }
    for (auto &t:Point_Circle) {
        for (int i=1; i<t.second.size(); ++i) {
            //圆的半径相同
LineEquation(Line(t.first,t.second[i-1]),Line(t.first,t.second[i]));
        }
        vector<Line> diameter;
        for (int i=0; i<t.second.size(); ++i) {
            for (int j=i+1; j<t.second.size(); ++j) {
                Line AB(t.second[i],t.second[j]);
                if
(isExist(AB)&&isCollinear(t.second[i],t.second[j],t.first))
diameter.push_back(AB);
            }
        }
        for (int i=0; i<diameter.size(); ++i) {
            for (int j=0; j<t.second.size(); ++j) {
                char C = t.second[j];
                char A = diameter[i].x;
                char B = diameter[i].y;
                if (C!=A&&C!=B) {
                    //点在圆上定理
                    AngleEquation(Angle(A,C,B),90);
                }
            }
            for (int j=i+1; j<diameter.size(); ++j) {
                //圆的直径相同
                LineEquation(diameter[i],diameter[j]);
            }
        }
    }
    //点在直线上
```

```
    for (auto &t:Point_Line) for (auto &p:t.second)
PointLiesOnLine(t.first,p);
    //两直线垂直
    for (auto &t:perpendicular) {
        Line A = t.first.first;
        Line B = t.first.second;
        if (A.x==B.x) AngleEquation(Angle(A.y,A.x,B.y),90);
        else if (A.x==B.y) AngleEquation(Angle(A.y,A.x,B.x),90);
        else if (A.y==B.x) AngleEquation(Angle(A.x,A.y,B.y),90);
        else if (A.y==B.y) AngleEquation(Angle(A.x,A.y,B.x),90);
        else {
            char o = getNode(A, B);
            AngleEquation(Angle(A.x,o,B.x),90);
            AngleEquation(Angle(A.x,o,B.y),90);
            AngleEquation(Angle(A.y,o,B.x),90);
            AngleEquation(Angle(A.y,o,B.y),90);
        }
    }
    //两直线平行
    for (auto &t:parallel) {
        Line A = t.first.first;
        Line B = t.first.second;
        char a = A.x, b = A.y;
        char c = B.x, d = B.y;
        for (auto &u:BaseLine) {
            Line C = u.first;
            if (C==A||C==B) continue;
            char x = getNode(A, C);
            char y = getNode(B, C);
            if (x==0||y==0) continue;
            if (a!=x&&c!=y) AngleEquation2(Angle(a,x,y),Angle(x,y,c));
            if (b!=x&&d!=y) AngleEquation2(Angle(b,x,y),Angle(x,y,d));
        }
    }
    REP(i,1,500) {
        solve();
        if (check()) return;
    }
    throw;
}

int main() {
    int t;
    cin>>t;
    while (t--) work();
}
```