game 20

Ели започна да пише първата си компютърна игра. В нея космически корабчета се обстрелват в 2D пространството със специални снаряди, които се движат по права линия и с времето стават по-големи (можете да си ги представите като разширяващи се кръгове, чиито център се движи по права линия). Когато някой от снарядите уцели базата на противника, на екрана се изписва „All your base are belong to us” и играта приключва. Ели не е сигурна дали е написала правилно модула, който проверява пресичането на снаряд с права, затова моли вас да й помогнете. Корабите са представени като неподвижни точки, които едновременно изстрелват по един снаряд. i-тият кораб се намира в точка (Xi, Yi). Снарядът на i-тия кораб изминава DXi единици в посока X и DYi единици в посока Y за единица време, като за това време разширява и радиуса си с DRi единици. Правата се задава чрез две различни точки по нея - (LX1, LY1) и (LX2, LY2).

Жокер: ОПИСАНАТА ТЕОРИЯ ЗА АНАЛИТИЧНА ФУНКЦИЯ НА ПРАВА Я НЯМА НИТО В "Програмиране = ++Алгоритми" нито на адрес www.inforatika.bg
Представете си (направо си го нарисувайте на хартия) момента на сблъсък на i-тия снаряд с линията L дефинирана от точките (LX1, LY1) и (LX2, LY2). Нека той да е след време Ti от началото на играта. В този момент снаряда е нараснал до кръг с радиус Ri = DRi * Ti, чийто център се намира вече в точката (Xi+DXi*Ti, Yi+DYi*Ti). Нашата линия се явява допирателна до окръжността представляваща периметъра на снаряда, следователно отсечката от точката на контакт до центъра на окръжността е перпендикулярна на линията L и дължината на тази отсечка DRi * Ti се явява най-краткото разстояние от центъра на кръга на снаряда до линията L.
Една права се изразява аналитично чрез нейната функция Fline(x, y) = a * x + b * y + c. Тук a, b и c се явяват изчислими коефициенти, които описват дадената права линия. Важна зависимост е че векторът с координати (a,b) е перпендикулярен (нормала) на линията (всеки неин колинеарен/успореден вектор). Тази аналитична функция на права ни дава още 3 ВАЖНИ зависимости:
1) Дадена точка с координати (Xт, Yт) лежи на дадена права Fline(x, y), тогава и само тогава, когато Fline(Xт, Yт) = 0 => a * Xт + b * Yт + c = 0. Това равенство се нарича уравнение на права.
2) За всички точки лежащи в едната полуравнина спрямо правата е изпълнено Fline(x, y) > 0, а за всички точки лежащи в другата полуравнина е изпълнено Fline(x, y) < 0. Това означава че ако имаме две точки T1(X1, Y1) и T2(X2, Y2) лежащи в една полуравнина то:
Sign(Fline(X1, Y1)) = Sign(Fline(X2, Y2)). Ако тези точки са в различни полуравнини то:
Sign(Fline(X1, Y1)) = -Sign(Fline(X2, Y2))
Тук Sign(X) = 1, ако X > 0 и Sign(X) = -1, ако X < 0.
3) Ако имаме точка с координати (Xт, Yт) и коефициентите a и b на аналитичната функция на правата представляват единичен вектор, т.е |(a, b)| = 1 или a*a+b*b=1, то |Fline(Xт, Yт)| ни дава най-краткото разстояние от тази точка до правата изразена чрез Fline(x, y). Зависимост 1) би била частен случай на тази, ако при тук нямахме ограничението |(a, b)| = 1. Под |X| имам предвид дължината на вектор X. Ако този вектор е едномерен (скаларна величина, каквато е Fline(Xт, Yт)), то |X| е абсолютната стойност на X - тя е просто частен случай на дължина на вектор, когато той е само с едно измерение.
Тази 3-та зависимост веднага ни насочва как да конструираме уравнението решаващо нашата задача. След време Ti центърът на нашият снаряд/кръг се намира в точка (Xi+DXi*Ti, Yi+DYi*Ti), която от своя страна е на разстояние Ri = DRi * Ti от правата L =>
УРАВНЕНИЕ 1: Fline(Xi+DXi*Ti, Yi+DYi*Ti) = Sign(Fline(Xi, Yi)) * DRi * Ti
Това означава, че: Fline(Xi+DXi*Ti, Yi+DYi*Ti) = +DRi * Ti, ако Fline(Xi, Yi) > 0
и
Fline(Xi+DXi*Ti, Yi+DYi*Ti) = -DRi * Ti, ако Fline(Xi, Yi) < 0

Тази условна зависимост произтича от зависимост 2) и факта че стартовата точка и центъра на снаряда/кръга в момента на сблъсъка лежат в една и съща полуравнина.
Остава само да намерим коефициентите a, b и c на нашата линия L. Ще пиша стрелката за вектор преди буквите му, тъй като тук не мога отгоре. Нека с ->COL обозначим колинеарния вектор на правата L дефинирана чрез точките L1(LX1, LY1) и L2(LX2, LY2). ->COL = ->L1L2 = ->L1O + ->OL2, където O(0,0) е центърът на координатната система => ->COL = - ->OL1 + ->OL2 = -(LX1, LY1) + (LX2, LY2) = (-LX1, -LY1) + (LX2, LY2) = (-LX1 + LX2, -LY1 + LY2) = (LX2-LX1, LY2-LY1). Неговият перпендикулярен вектор се конструира като се разменят координатите X и Y и се постави минус пред едната. Примерно, ако имаме вектора (1, 2), то неговите перпендикулярни вектори са (2, -1) и (-2, 1). Начертайте си го и измерете с транспортир, ако не вярвате. Следователно PERP_VEC(->COL) = (LY2-LY1, -(LX2-LX1)) = (LY2-LY1, LX1-LX2). От по горната теория знаем че координатите на PERP_VEC(->COL) са коефициентите a и b на нашата права L => a = LY2-LY1 и b = LX1-LX2. За да е изпълнена зависимост 3) трябва (a,b) да е единичен вектор. Нормализираме a и b, разделяйки ги на дължината на вектора(a,b). Дължината на даден вектор (X,Y) намираме по питагорова теорема: |(X,Y)| = sqrt(X*X + Y*Y). Ако векторът е n-мерен, то: |(v1, v2, ... , vn)| = sqrt(v1 * v1 + v2 * v2 + ... + vn*vn).
Получаваме:
ново нормализирано a = (старо ненормализирано a)/|PERP_VEC(COL)| = (LY2-LY1)/sqrt((LY2-LY1) * (LY2-LY1) + (LX1-LX2)*(LX1-LX2))
и
ново нормализирано b = (старо ненормализирано b)/|PERP_VEC(COL)| = (LX1-LX2)/sqrt((LY2-LY1) * (LY2-LY1) + (LX1-LX2)*(LX1-LX2))
Имайки вече новите нормализирани a и b заместваме първата точка от правата (LX1, LY1) (няма проблем да заместите и втората - ще получите същата числена стойност за c) в зависимост 1) и така получаваме c:
a * LX1 + b * LY1 + c = 0 =>
=> c = -(a * LX1 + b * LY1)
Имайки вече a, b и c вече можем да решим УРАВНЕНИЕ 1 за i-тия снаряд. Пресмятаме Sign(Fline(Xi, Yi)) = Sign(a * Xi + b * Yi + c). Ако той е положителен (1) решаваме:
Fline(Xi+DXi*Ti, Yi+DYi*Ti) = +DRi * Ti =>
a*(Xi+DXi*Ti) + b*(Yi+DYi*Ti) + c = +DRi * Ti =>
a*Xi+a*DXi*Ti + b*Yi+b*DYi*Ti + c = +DRi * Ti =>
+DRi * Ti - a*DXi*Ti - b*DXi*Ti = a*Xi + b*Yi + c =>
Ti * (DRi - a*DXi - b*DYi) = a*Xi + b*Yi + c =>
Ti = (a*Xi + b*Yi + c) / (DRi - a*DXi - b*DYi)
Ако (DRi - a*DXi - b*DYi) ще имаме деление на нула. Това е случая когато снаряда никога няма да докосне линията. Така че няма да имаме валидно Ti.
Ако получим отрицателно Ti означава че снарядът би трябвало да се движи наобратно за да удари линията, така че и тук Ti е невалидно.

Ако Sign(a * Xi + b * Yi + c) е отрицателен (-1) решаваме:
Fline(Xi+DXi*Ti, Yi+DYi*Ti) = -DRi * Ti =>
... =>
Ti = -(a*Xi + b*Yi + c) / (DRi + a*DXi + b*DYi) Отново, ако знаменателя DRi + a*DXi + b*DYi == 0 или Ti < 0, то Ti е невалидно.

По този начин пресмятаме Ti за всички снаряди. Ако всички са невалидни печатаме NEVER, в противен случай печатаме min(Ti) за тези i = 1..N при които Ti е валидно.
За да отпечатате число закръглено до n-тия знак след десетичната запетая ви трябва:
#include <iostream>

```
#include <iomanip> // YOU NEED THIS FOR std::fixed and std::setprecision(...)
...
int n=...
double number =...
...
std::cout << std::fixed << std::setprecision(n) << number << std::endl;
```

Ако нямате std::fixed ще се отпечатат общо n цифри от числото без значение колко от тях са преди или след десетичната запетая.

Input Format

На първия ред на стандартния вход ще бъде зададен броят тестове T. Всеки от тях ще започва с ред, на който е даден броят кораби N. На следващите N реда ще бъдат зададени по пет цели числа Xi, Yi, DXi, DYi, DRi – съответно координатите на кораба и информация за снарядите му. Всеки тест завършва с ред, съдържащ 4 цели числа LX1, LY1, LX2, LY2 – две точки по правата, с която Ели иска да пресече снарядите. Гарантирано е, че няма да има кораб, който да лежи на правата.

Constraints

```
1 ≤ T ≤ 50
1 ≤ N ≤ 1000
0 ≤ DRi ≤ 10,000
-10,000 ≤ Xi, Yi, DXi, DYi, LX1, LY1, LX2, LY2 ≤ 10,000
```

Output Format

За всеки тест изведете по един ред с едно единствено реално число, закръглено до точно 3 знака след десетичната точка – времето до първото пресичане на снаряд с правата. Ако това никога не се случва, изведете NEVER.

Sample Input 0

```
3
2
1 9 0 -3 2
4 1 -1 1 1
1 3 3 4
1
1 1 1 1 1
0 0 1 -1
3
42 42 13 17 3
2 42 1 -4 2
113 117 -1 -2 7
1 -1 -1 1
```

Sample Output 0

```
1.146
NEVER
7.549
```