130. Прямокутник

Дано координати трьох точок, вершин прямокутника. Знайдіть координати четвертої точки.

Вхідні дані

В єдиному рядку записано шість чисел координати трьох точок.

Вихідні дані

Два числа, координати шуканої вершини прямокутника. Всі вхідні та вихідні дані - цілі числа по модулю не перевищують 100.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

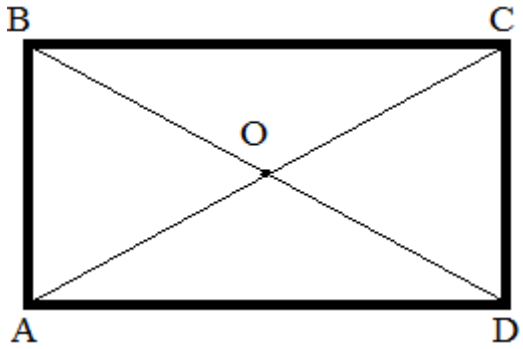
Вхідні дані

0 0 0 1 2 1

Вихідні дані

2 0

Розв’язання



Знайдемо у якій із трьох заданих вершин прямокутника є прямим, перевіривши всі можливі скалярні добутки. Пам’ятаємо, що коли рівняння (8) з теоретичного матеріалу дорівнює 0, то кут – прямий. Використаємо координати двох інших вершин для знаходження координат точки перетину діагоналей прямокутника, застосовуючи формули (2) із теоретичного матеріалу при k=1. Знайдемо невідомі координати вершини прямокутника з формул (2), наприклад:

$$p_1 = 2x_0 - x_1, p_2 = 2y_0 - y_1.$$

При розв’язанні слід пам’ятати про те, що координати вершин прямокутника – цілі числа.

Код програми:

```
var x1,x2,x3,x4,y1,y2,y3,y4,xa,ya,xb,yb,xc,yc,p1,p2:integer;
    y0,x0:real;
function ScalProd(x1, y1, x2, y2:integer):integer;
begin
  ScalProd:= x1* x2+ y1*y2
 end;
begin
 readln(x1,y1,x2,y2,x3,y3);
 xa:=x2-x1; ya:=y2-y1;
 xb:=x3-x1; yb:=y3-y1;
 xc:=x3-x2; yc:=y3-y2;
 if ScalProd(xa,ya,xb,yb)=0 then
  begin
```

```
   x0:=(x2+x3)/2; y0:=(y2+y3)/2;
   p1:=round(2*x0-x1);  p2:=round(2*y0-y1);
  end;
 if ScalProd(xa,ya,xc,yc)=0 then
  begin
   x0:=(x1+x3)/2; y0:=(y1+y3)/2;
   p1:=round(2*x0-x2);  p2:=round(2*y0-y2);
  end;
 if ScalProd(xc,yc,xb,yb)=0 then
  begin
   x0:=(x2+x1)/2; y0:=(y2+y1)/2;
   p1:=round(2*x0-x3);  p2:=round(2*y0-y3);
  end;
  writeln(p1,' ',p2);
end.
```

359. Коло і пряма

Є коло радіуса **R** з координатами центра **x,y** і пряма, що задана координатами двох своїх точок. Якої довжини відрізок прямої лежить всередині кола?

Вхідні дані

Задано рядок з **7**-ми чисел: радіус кола, координати **x, y** центра і координати **2**-х точок прямої. Всі числа цілі, за абсолютним значенням не перевищують **10000**.

Вихідні дані

Вивести шукану довжину з точністю до **5** цифр після коми. Якщо коло і пряма не перетинаються, вивести **-1**, якщо дотикаються - вивести **0**.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** МіВ

Вхідні дані

5 0 0 4 1 4 2

Вихідні дані

6.00000

Розв’язання

Взаємне розташування кола і прямої подано на рисунку 8 теоретичного матеріалу. Знаходимо відстань від центру кола до прямої, яка задана двома точками. Якщо відстань більша за радіус, то пряма і коло не перетинаються, коли ж рівні, то дотикаються, а інакше – слід знайти довжину хорди, яка сполучає дві точки перетину прямої і кола. Враховуємо, що відстань обчислена наближено. Для знаходження точок перетину прямої і кола – використаємо формулу (25) теоретичного матеріалу.

Код програми:

```
var r,x0,y0,x1,y1,x2,y2:integer;
    a,b,c,aa,bb,cc,x3,y3,x4,y4,l,d,d1:real;
begin
 readln(r,x0,y0,x1,y1,x2,y2);
 l:=abs((x2-x1)*(y0-y1)-(y2-y1)*(x0-x1) )/sqrt(sqr(x2-x1)+sqr(y2-
y1));
 if (l-r)>0.00001 then writeln(-1)
```

```
 else
 if abs(l-r)<=0.00000005 then writeln(0)
 else
  begin
   a:=y2-y1;
   b:=x1-x2;
   c:=x1*(y1-y2)+y1*(x2-x1);
   if abs(b)<=0.000000001 then
    begin
     y3:=-sqrt(r*r-sqr(c/a+x0))+y0;
     y4:=sqrt(r*r-sqr(c/a+x0))+y0;
     x3:=-c/a;
     x4:=-c/a;
     d:=sqrt(sqr(x4-x3)+sqr(y4-y3));
     writeln(d:0:5);
    end
    else
     begin
       aa:=a*a/(b*b)+1;
       bb:=(y0+c/b)*a/b-x0;
       cc:=x0*x0+sqr(y0+c/b)-r*r;
       d1:=bb*bb-aa*cc;
       x3:=(-bb-sqrt(d1))/aa;
       x4:=(-bb+sqrt(d1))/aa;
       y3:=(-a*x3-c)/b;
       y4:=(-a*x4-c)/b;
       d:=sqrt(sqr(x4-x3)+sqr(y4-y3));
       writeln(d:0:5);
     end;
  end;
end.
```

839. Перетин відрізків

Два відрізки на площині задано цілочисельними координатами своїх кінців у декартовій системі координат.

Потрібно визначити, чи існує у них спільна точка.

Вхідні дані

У першому рядку містяться координати першого кінця першого відрізка, у другому - другого кінця першого відрізка, у третьому і четвертому - координати кінців другого відрізка. Координати цілі і по модулю не перевищують **10000**.

Вихідні дані

Вивести слово "**Yes**", якщо спільна точка є, або слово "**No**" у протилежному випадку.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

Sample 1

0 0

1 0

1 0
1 1

Sample 2
0 0
1 0
2 0
3 0

Вихідні дані

Sample 1
Yes

Sample 2
No

Розв’язання

Дана задача є базовою задачею 8 теоретичного матеріалу.

Код програми:

```
var xx1,yy1,xx2,yy2,xx3,yy3, xx4, yy4:real;
function SlantMult(x1,y1,x2,y2:real):real;
 begin
  SlantMult:=x1*y2-x2*y1;
 end;
function SectionSegments(x1, y1, x2, y2, x3, y3, x4,
y4:real):boolean;
var b:boolean;
    xmax1,xmax2,xmin1,xmin2,ymax1,ymax2,ymin1,ymin2,acx,acy,abx,
    aby, adx, ady, cax, cay, cdx, cdy, cbx, cby:real;
 begin
  if x1>x2 then
   begin
    xmax1:=x1;
    xmin1:=x2;
   end
  else
   begin
    xmax1:=x2;
    xmin1:=x1;
   end;
   if x3>x4 then
   begin
    xmax2:=x3;
    xmin2:=x4;
   end
  else
   begin
    xmax2:=x4;
    xmin2:=x3;
   end;
```

```
   if y1>y2 then
   begin
    ymax1:=y1;
    ymin1:=y2;
   end
  else
   begin
    ymax1:=y2;
    ymin1:=y1;
   end;
   if y3>y4 then
   begin
    ymax2:=y3;
    ymin2:=y4;
   end
  else
   begin
    ymax2:=y4;
    ymin2:=y3;
   end;
  if
(xmax1>=xmin2)and(xmax2>=xmin1)and(ymax1>=ymin2)and(ymax2>=ymin1)
then
  begin
   acx:=x3-x1; acy:=y3-y1;
   abx:=x2-x1; aby:=y2-y1;
   adx:=x4-x1; ady:=y4-y1;
   cax:=x1-x3; cay:=y1-y3;
   cbx:=x2-x3; cby:=y2-y3;
   cdx:=x4-x3; cdy:=y4-y3;
  b:=(SlantMult(acx,acy,abx,aby)*SlantMult(adx,ady,abx,aby)<=0)
and
  ((SlantMult(cax,cay,cdx,cdy)*SlantMult(cbx,cby,cdx,cdy)<=0));
  if b then SectionSegments:=true
    else SectionSegments:=false;
  end
   else  SectionSegments:=false;
 end;
begin
 readln(xx1,yy1);
 readln(xx2,yy2);
 readln(xx3,yy3);
 readln(xx4,yy4);
 if SectionSegments(xx1, yy1, xx2, yy2, xx3, yy3, xx4, yy4)
 then writeln('Yes')
 else writeln('No');
 end.
```

1265. Єгипет

Ще у давні часи єгиптяни зрозуміли, що трикутник зі сторонами **3**, **4** і **5** має прямий кут, який є його найбільшим кутом. Ви повинні визначити чи й інші трикутники мають цю ж властивість.

Вхідні дані

Вхідні дані містять декілька наборів тестових даних, які завершуються рядком, що містить **000**. Кожен тестовий випадок містить три натуральних числа, які не перевищують **30000** і задають сторони чергового трикутника.

Вихідні дані

Для кожного тестового випадку в окремому рядку виведіть "**right**" якщо трикутник є прямокутним або "**wrong**" у протилежному випадку.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

6 8 10
25 52 60
5 12 13
0 0 0

Вихідні дані

right
wrong
right

Розв’язання

Для розв’язання даної задачі слід визначити, яка із довжин сторін трикутника є найбільшою у кожному з випадків. Далі слід перевірити чи виконується теорема Піфагора. Якщо виконується – вивести слово `'right'`, а інакше – слово `'wrong'`.

Код програми:

```
var a,b,c,k:int64;
begin
 readln(a,b,c);
 while (a<>0)and(b<>0)and(c<>0) do
 begin
  if c<a then
  begin
   k:=c;
   c:=a;
   a:=k;
   if c<b then
  begin
   k:=c;
   c:=b;
   b:=k;
  end;
  end;
  if c<b then
```

```
  begin
    k:=c;
    c:=b;
    b:=k;
    if c<a then
  begin
    k:=c;
    c:=a;
    a:=k;
    end;
  end;
  if (a*a+b*b=c*c) then writeln('right')
else
  if (a*a+b*b<>c*c) then  writeln('wrong');
  readln(a,b,c);
 end;
end.
```

1657. Прямокутники

Задано дійсні числа **a**, **b**, **c**, **d**.

Вияснити, чи можна у прямокутник зі сторонами **a**, **b** цілком помістити прямокутник зі сторонами **c**, **d**.

Вхідні дані

У першому рядку вхідного файлу знаходяться дійсні числа **a**, **b**, **c** та **d**, відокремлені одним чи декількома пропусками або символами переведення рядка.

Вихідні дані

Виведіть у вихідний файл слово **YES**, якщо другий прямокутник можна помістити у перший, або слово **NO** у протилежному випадку.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

Sample 1

317 10 11 23

Sample 2

31 10 10 31

Вихідні дані

Sample 1

NO

Sample 2

YES

Розв’язання

Уведемо позначення H_k – висота конверта, W_k – ширина конверта, H_l – висота листівки, W_l – ширина листівки. Відомо, що при цьому повинні виконуватися умови: $H_k \le W_k$, $H_l \le W_l$. Тому зчитані чотири числа відсортуємо спочатку так, щоб виконувалися вказані нерівності. Зробити це досить просто – якщо нерівність для двох відповідних чисе не виконується, то потрібно обміняти значення двох змінних.

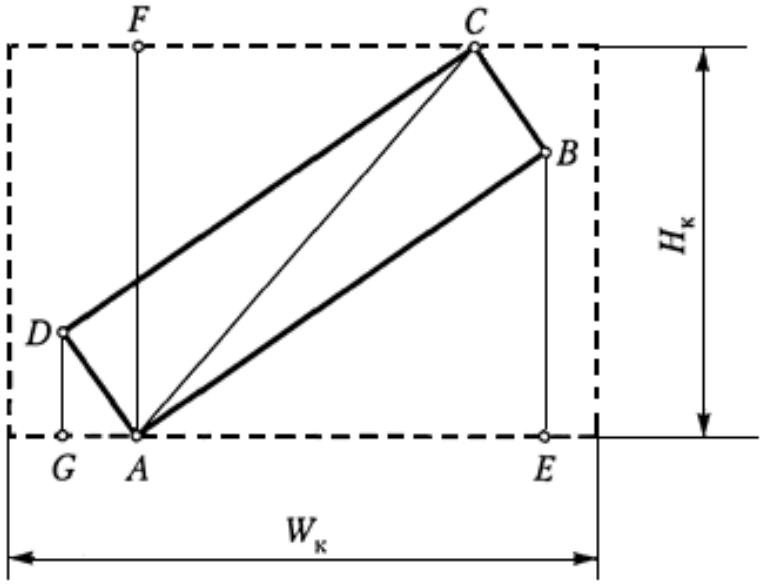


Рис. 16

Якщо виконується умова $H_l > H_k$, то листівку помістити в конверт не можна. Це випливає з того, що в листівку поміститься коло діаметром H_l, а в конверт коло такого діаметру не поміститься. Значить, не поміститься і листівка.

Розглянемо випадок, коли $H_l \le H_k$. За умови, що виконується нерівність $W_l \le W_k$, можна зробити висновок – листівка входить у конверт паралельно сторонам конверта. І залишається дослідити більш складний випадок: $H_l < H_k$ і $W_l > W_k$. У такому випадку можна спробувати розмістити листівку в конверт не паралельно сторонам конверта. Листівку можна розмістити лише в тому випадку, коли один її кут буде розташований на нижній основі конверта, а другий кут – на верхній основі. Спробуємо підняти листівку на максимальний кут від горизонту і визначити чи поміститься вона в конгверт.

Нехай прямокутник ABCD – листівка (див. рис.16), точка А якої лежить на нижній основі конверта, а точка С – на верхній його основі. На рисунку $\angle A = \angle B = \angle C = \angle D = \frac{\pi}{2}, AB = CD = W_l, BC = AD = H_l, AF = H_k$.

Щоб визначити чи поміститься листівка в конверті, потрібно знайти довжину GE та порівняти її з W_k. Маємо $GE = GA + AE$ (41). З ΔAGD ($\angle G = \frac{\pi}{2}$) маємо $GA = ADsin\angle GDA$. З ΔABE ($\angle E = \frac{\pi}{2}$) маємо $AE = ABcos\angle BAE$. Отже, слід знайти $\angle GDA$ і $\angle BAE$.

З прямокутного трикутника АВС, за теоремою Піфагора, знаходимо

$AC = \sqrt{AB^2 + BC^2} = \sqrt{W_l^2 + H_l^2}$. Аналогічно з прямокутного трикутника ACF, використовуючи теорему Піфагора, знаходимо $CF = \sqrt{AC^2 - AF^2} = \sqrt{W_l^2 + H_l^2 - H_k^2}$.

З ΔACB ($\angle B = \frac{\pi}{2}$) знайдемо $\angle CAB = arctg\frac{BC}{AB}$, $\angle BAC = \frac{\pi}{2} - \angle CAB$. Аналогічно з ΔAFC ($\angle F = \frac{\pi}{2}$) знайдемо $\angle FAC = arctg\frac{FC}{AF}$, $\angle CAF = \frac{\pi}{2} - \angle FAC$.

Оскільки $DG \perp GE, AF \perp GE$, то $DG \parallel AF$. Для прямих DG і AF пряма DA буде січною, тому $\angle GDA = \angle DAF$ – як внутрішні різносторонні кути. З рис. 16 видно, що $\angle BAF = \angle BAC + \angle CAF$. Тоді з $\angle BAE = \frac{\pi}{2} - \angle BAF$, $\angle DAF = \frac{\pi}{2} - \angle BAF$, маємо $\angle GDA = \frac{\pi}{2} - \angle BAF$.

Знайшовши все необхідне, підставляємо у формулу (41) і обчислюємо GE. Якщо $GE < W_k$, то листівка поміститься в конверті, а інакше – ні.

Залишилося лише написати код на мові програмування Free Pascal. Визначимося щодо типів, які слід використовувати. Здавалося, що можна перевести розміри листівки й конверта у міліметри й використовувати один із цілочисельних діапазонів. Проте обчислення кореня квадратного й тригонометричних функцій все одно приводить до необхідності використання дійсного типу, наприклад, extended. Щоб усунути можливі помилки в обчисленнях, будемо використовувати порівняння не з нулем, а з константою eps=1e-10.

Якщо розглядати тільки розміщення листівки горизонтально чи вертикально до сторін конверта, то проходить 82% тестів.

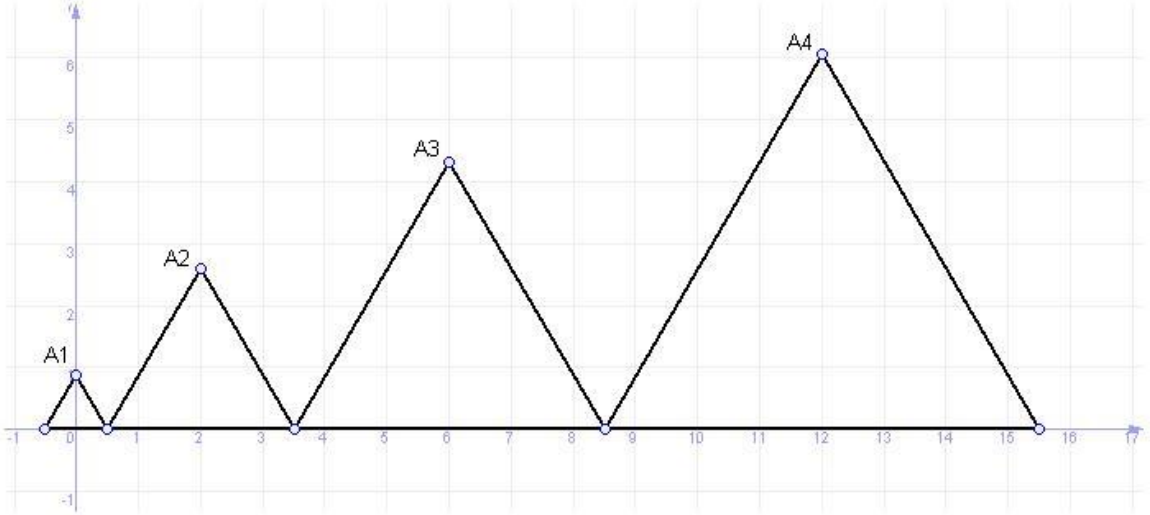
Код програми:

```
{$N+}
const eps=1e-10;
var a,b,c,d,hk,wk,hl,wl,ab,cd,bc,ad,af,ge,ga,ae,ac,cf,
    bae,gda,baf,bac,caf,cab,fac:extended;
    s:string;
begin
 readln(a,b,c,d);
 if a<b then
  begin
   wk:=b;
   hk:=a;
  end
 else
 begin
   wk:=a;
   hk:=b;
 end;
 if c<d then
  begin
   wl:=d;
   hl:=c;
  end
 else
 begin
```

```
   wl:=c;
   hl:=d;
 end;
  if hl>hk then s:='NO'
  else
   begin
    if wk>=wl then s:='YES'
     else
      begin
       ab:=wl; cd:=wl;
       bc:=hl; ad:=hl;
       af:=hk;
       ac:=sqrt(wl*wl+hl*hl);
       cf:=sqrt(ac*ac-hk*hk);
       cab:=arctan(bc/ab);
       fac:=arctan(cf/af);
       bae:=pi/2-cab-fac;
       ge:=ad*sin(bae)+ab*cos(bae);
       if (wk-ge)>=eps then s:='YES' else s:='NO';
      end;
   end;
   writeln(s);
end.
```

2066. Васині трикутники

Вася побудував рівносторонній трикутник зі стороною, рівною **1**, назвав його $\mathbf{A_1B_1C_1}$ і розмістив одну зі сторін на осі **OX** так, що її середина розмістилась точно у початку координат, а вершина $\mathbf{A_1}$ виявилась розміщеною на додатній півосі **OY**. Кожен наступний Васин рівносторонній трикутник має довжину сторони рівно на **2** більшу і розміщений правіше попереднього так, що одна з вершин співпадає на осі **OX** з крайньою правою вершиною попереднього трикутника, а вершина $\mathbf{A_n}$ розміщена у верхній півплощині.



Тепер Васю цікавить питання: *які координати має вершина* $\mathbf{A_n}$ **n**-*го трикутника*? Допоможіть йому.

Вхідні дані

У єдиному рядку вхідного файлу задано номер трикутника **n** ($\mathbf{1 \le n \le 10^4}$).

Вихідні дані

У єдиному рядку вихідного файлу виведіть з трьома знаками після коми координати вершини $\mathbf{A_n}$ $\mathbf{X_n}$ і $\mathbf{Y_n}$ трикутника, який цікавить Васю, розділивши їх одним пропуском.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

2

Вихідні дані

2.000 2.598

Розв’язання

З умови задачі випливає, що при заданні довжин сторін правильного трикутника використовується арифметична прогресія. Легко помітити, що при знаходженні X_n координати слід знайти суму n членів арифметичної прогресії, але від неї слід відняти $\frac{1}{2}$ (через зміщення першого трикутника ліворуч від початку координат) і $\frac{a_n}{2}$ (бо вершина трикутника проектується на середину сторони). Для знаходження Y_n слід знайти висоту відповідного правильного трикутника, довжина сторони якого a_n.

Код програми:

```
var an,a1,d,n:qword;
    x,y:real;
begin
 readln(n);
 a1:=1;
 d:=2;
 an:=a1+d*(n-1);
 x:=(a1+an)*n/2-0.5-an/2;
 y:=sqrt(3)*an/2;
 writeln(x:0:3,' ',y:0:3);
end.
```

2146. Бісектриса

Знайдіть пряму, яка містить бісектрису кута, заданого вершиною X та двома точками Y та Z на його сторонах.

Вхідні дані

Шість чисел - координати точок X, Y та Z. Усі вхідні дані цілі числа, які не перевищують по модулю 10000. Кут гарантовано більший 0 і менший 180 градусів.

Вихідні дані

Три числа - коефіцієнти загального рівняння бісектриси кута ∠YXZ. Коефіцієнти виводьте з точністю три знаки після коми.

Ліміт часу 1 секунда

Ліміт використання пам'яті 64 MiB

Вхідні дані

1 1 1 0 0 1

Вихідні дані

-1.000 1.000 0.000

Розв’язання

Для розв’язання цієї задачі слід використати формули (16) і (21).

Код програми:

```
var x0,y0,x1,y1,x2,y2,px1,py1,px2,py2,a,b,c,k:real;
begin
 readln(x0,y0,x1,y1,x2,y2);
 px1:=x1-x0; py1:=y1-y0;
 px2:=x2-x0; py2:=y2-y0;
 a:=py1/sqrt(px1*px1+py1*py1)+py2/sqrt(px2*px2+py2*py2);
 b:=-(px1/sqrt(px1*px1+py1*py1)+px2/sqrt(px2*px2+py2*py2));
 c:=-x0*a-y0*b;
 if abs(a)=abs(b) then
  begin
   k:=abs(a);
   a:=a/k;
   b:=b/k;
   c:=c/k;
  end;
 writeln(a:0:3,' ',b:0:3,' ',c:0:3);
end.
```

2147. Площа многокутника

Визначити площу заданого многокутника.

Вхідні дані

У першому рядку одне число - **N** (**3** ≤ **N** ≤ **100000**). Далі у **N** рядках по парі чисел - координати чергової вершини простого многокут у порядку обходу за чи проти годинникової стрілки. Усі координати цілі числа, які по модулю не перевищують **10000**.

Вихідні дані

Одне число - площа заданого многокутника з точністю **1** знак після коми.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

3
1 0
0 1
1 1

Вихідні дані

0.5

Розв’язання

Для розв’язання даної задачі слід застосувати формулу (34) теоретичного матеріалу.

Код програми:

```
{$N+}
var x,y,y1:array [1..100001] of longint;
    n,i,j,k,ymin:longint;
    s:extended;
begin
 readln(n);
```

```
 for i:=1 to n do readln(x[i],y[i]);
 x[n+1]:=x[1];
 y[n+1]:=y[1];
 ymin:=y[1];
 for i:=2 to n do
  if ymin>y[i] then ymin:=y[i];
 for i:=1 to n+1 do
  y1[i]:=y[i]-ymin;
 s:=0;
 for k:=1 to n do
 s:=s+(x[k+1]-x[k])*(y1[k+1]+y1[k]);
 s:=abs(s)/2;
 writeln(s:0:1);
end.
```

2148. Опуклий многокутник

Визначити, чи є заданий многокутник опуклим.

Вхідні дані

У першому рядку одне число - **N** (**3** ≤ **N** ≤ **100000**). Далі у **N** рядках по парі чисел - координати чергової вершини простого многокутника у порядку обходу за або проти годинникової стрілки. Усі координати цілі числа, які по модулю не перевищують **10000**.

Вихідні дані

Один рядок "**YES**", якщо наведений многокутник є опуклим, і "**NO**" у протилежному випадку.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

3
1 0
0 1
1 1

Вихідні дані

YES

Розв’язання

Дивись розв’язання базової задачі 12.

Код програми:

```
var n,i,j,k,m:longint;
    x,y:array [1..100000] of integer;
    l:string;
    z1,z2:int64;
begin
 readln(n);
 for i:=1 to n do
  readln(x[i],y[i]);
 l:='YES';
 for i:=1 to n do
 begin
```

```
  j:=(i mod n)+1;
  k:=(j mod n)+1;
  m:=i-1;
  if i=1 then m:=n;
  z1:=(x[m]-x[i])*(y[j]-y[i])-(y[m]-y[i])*(x[j]-x[i]);
  z2:=(x[k]-x[i])*(y[j]-y[i])-(y[k]-y[i])*(x[j]-x[i]);
  if z1*z2<0 then
   begin
    l:='NO';
    break;
   end;
 end;
 writeln(l);
end.
```

4782. Точка перетину прямих

На площині задано дві прямі. Кожна пряма задається парою точок, через які вона проходить.

Потрібно встановити, чи перетинаються ці прямі, та знайти координати точки перетину.

Вхідні дані

Вводяться спочатку координати двох різних точок, через які проходить перша пряма, а потім - координати ще двох різних (еле, можливо, спввпадаючих з першими двома) точок, через які проходить друга пряма. Координати кожної точки - цілі числа, які по модулю не перевищують **1000.**

Вихідні дані

Якщо прямі не перетинаються, виведіть одне число **0**. Якщо прямі співпадають, виведіть **2**. Якщо прямі перетинаються рівно в одній точці, то виведіть спочатку число **1**, а потім два дійсних числа - координати точки перетину.

Ліміт часу **1** секунда

Ліміт використання пам'яті **64** MiB

Вхідні дані

Sample 1

1 1 2 2

1 10 2 11

Sample 2

0 0 1 1

1 0 -1 2

Вихідні дані

Sample 1

0

Sample 2

1 0.5 0.5

На час написання методичного посібника даний розв’язок увійшов до числа 10 кращих розв’язань задачі. Проте при зміні тестів відбувся збій і на час проведення контесту перевірка проводилася не правильно.

Записуємо рівняння прямих у загальному вигляді:

$$\begin{cases} a_1x + b_1y + c_1 = 0; \\ a_2x + b_2y + c_2 = 0. \end{cases}$$

Застосовуємо метод Крамера до розв’язування даної системи:

$$d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1 \cdot b_2 - a_2 \cdot b_1;$$

$$d_x = \begin{vmatrix} -c_1 & b_1 \\ -c_2 & b_2 \end{vmatrix} = -c_1 \cdot b_2 - (-c_2) \cdot b_1;$$

$$d_y = \begin{vmatrix} a_1 & -c_1 \\ a_2 & -c_2 \end{vmatrix} = a_1 \cdot (-c_2) - a_2 \cdot (-c_1).$$

Якщо одночасно d, d_x і d_y дорівнюють 0, то прямі співпадають (виводимо 2), якщо $d = 0$, а хоча б одне з d_x чи d_y не дорівнює 0 – прямі не перетинаються (виводимо 0), інакше обчислюємо координати $x := \frac{dx}{d}$, $y := \frac{dy}{d}$.

Код програми:

```
var x1,y1,x2,y2,x3,y3,x4,y4,a1,b1,c1,a2,b2,c2,x,y,d,dx,dy:real;
begin
 readln(x1,y1,x2,y2);
  readln(x3,y3,x4,y4);
 a1:=y1-y2; b1:=x2-x1; c1:=x1*(y2-y1)-y1*(x2-x1);
 a2:=y3-y4; b2:=x4-x3; c2:=x3*(y4-y3)-y3*(x4-x3);
 d:=a1*b2-a2*b1;
 dx:=(-c1)*b2-(-c2)*b1;
 dy:=a1*(-c2)-a2*(-c1);
 if d=0 then
 begin
  if (dx=0)and(dy=0) then writeln(2) else writeln(0);
 end
 else
  begin
  x:=dx/d;
  y:=dy/d;
  if x=-0.00 then x:=0;
  if y=-0.00 then y:=0;
 writeln('1 ',x:0:1,' ',y:0:1);
 end;
end.
```