Lab 12/05

1. A complex number is a number that contains two parts: a real part and an 'imaginary' part. The real part is any real number. The imaginary part is a real number multiplied by the imaginary unit written as the lower-case letter 'i'. Complete the following functions.

```
void C_print (double a, double b)
{
// print out (a + bi) or (a - bi) according to the sign of b
// print out a if b==0
}

void C_add (double a, double b , double c, double d, double *e , double *f)
{
// (e+fi) = (a+bi)+(c+di)
}

void C_sub (double a, double b , double c, double d, double *e , double *f)
{
// (e+fi) = (a+bi)-(c+di)
}
void C_mul (double a, double b , double c, double d, double *e , double *f)
{
// (e+fi) = (a+bi)*(c+di)
}
void C_div (double a, double b , double c, double d, double *e , double *f)
{
// (e+fi) = (a+bi)/(c+di)
}
```

Write a main function to test your subroutine. The part of main function should be like this:

```
int main()
{
    ………… // Something you should implement.

    // The output should be like: (5 + 3i) + (4 - 3i) = 9
    C_print(…………);
    printf(…………);
    C_print(…………);
    C_add(…………);
    printf(…………);
    C_print(…………);
    printf(“\n”);

    ………… // Something you should implement.

    return 0;
}
```

Your output should look like this:

```
(5 + 3i) + (4 - 3i) = 9
(5 + 3i) - (4 - 3i) = (1 + 6i)
(5 + 3i) * (4 - 3i) = (29 - 3i)
(5 + 3i) / (4 - 3i) = (0.44 + 1.08i)
```

FYI:

$$(a+bi)+(c+di)=(a+c)+(b+d)i$$

$$(a+bi)\times(c+di)=a\times c+a\times di+c\times bi+bi\times di=(ac-bd)+(ad+bc)i$$

$$\frac{a+bi}{c+di}=\frac{(a+bi)(c-di)}{(c+di)(c-di)}=\frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

2. Please follow the requirements to implement the functions and write a main function to test your result. You can make use of math functions in math.h.

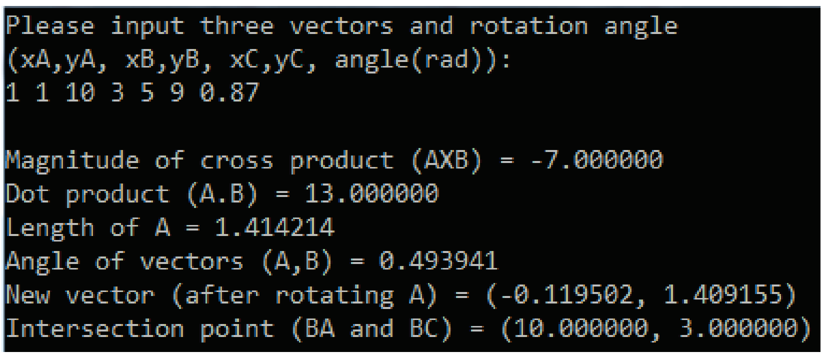
Function name	Description	Parameters and What to Return
cross	Calculate the magnitude of cross product of two vectors (this can be interpreted as the positive area of the parallelogram) EX: **u**=(x_1, y_1), **v**=(x_2, y_2) **u** x **v** = x_1y_2 - y_1x_2	1. Four parameters for two vectors, each vector has two arguments. 2. Return magnitude of cross product of 2D vectors.
dot	Calculate dot product of two vectors EX: **u**=(x_1, y_1), **v**=(x_2, y_2) **u** • **v** = x_1x_2 + y_1y_2	1. Four parameters for two vectors, each vector has two arguments. 2. Return dot product.
length	Calculate length of one vector EX: **u**=(x, y) $\lVert\mathbf{u}\rVert=\sqrt{x^2+y^2}$	1. Two parameters for one vector. 2. Return length of vector.
angle	Calculate angle between two vector EX: **u**=(x_1, y_1), **v**=(x_2, y_2) $\cos\theta=\frac{\mathbf{u}\bullet\mathbf{v}}{\lVert\mathbf{u}\rVert\lVert\mathbf{v}\rVert}\Rightarrow\theta=\arccos\theta$	1. Four parameters for two vectors, each vector has two arguments. 2. Return angle between two vectors.
rotate	Rotate one vector with an angle, the rotation matrix to rotate the vector is as follows. EX: **u**=(x, y), θ is rotation angle	1. Two parameters for one vector, one parameter for rotation angle (the unit is radius). 2. Use call by address to return two parameters for rotated vector. You can refer to

	$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$	
getLineIntersection	Get intersection point between two lines. You can refer to following equation to obtain the intersection point. Two lines: (**p** + *t* x **v**) and (**q** + *s* x **w**), where **p** and **q** are points, **v** and **w** are vectors, *t* and *s* are scalar parameters. So you can obtain $t = \frac{\mathbf{w}\times(\mathbf{p}-\mathbf{q})}{\mathbf{v}\times\mathbf{w}}$ or $s = \frac{\mathbf{v}\times(\mathbf{p}-\mathbf{q})}{\mathbf{v}\times\mathbf{w}}$ for the intersection point (*cross* is magnitude of cross product).	1. Eight parameters for two lines with the form of parametric equation of line (two parameters are for one point coordinate and two parameters for one vector, there are two lines). 2. Use call by address to return two parameters for intersection point.

The following code is part of testing main function, please complete it.

```
int main()
{
    double xa, ya, xb, yb, xc, yc, ang; // three vectors and one angle
    double xr, yr; // the vector after rotating

    printf("Please input three vectors and rotation angle\n");
    printf("(xA,yA, xB,yB, xC,yC, angle(rad)):\n");
    scanf("%lf%lf%lf%lf%lf%lf%lf", &xa, &ya, &xb, &yb, &xc, &yc, &ang);

    printf("\n");
    printf("Magnitude of cross product (AXB) = ......);
    printf("Dot product (A.B) = ......);
    printf("Length of A = ......);
    printf("Angle of vectors (A,B) = ......);

    rotate(......);
    printf("New vector (after rotating A) ......);

    getLineIntersection(......);
    printf("Intersection point (BA and BC) = ......);

    return 0;
}
```

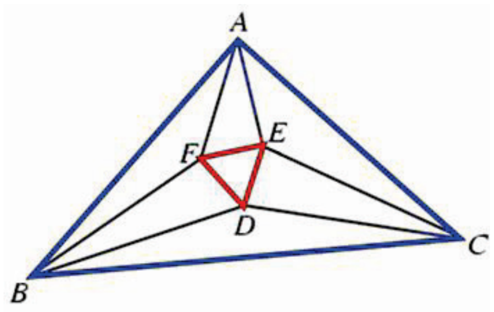
And you should obtain output like this:



```
Please input three vectors and rotation angle
(xA,yA, xB,yB, xC,yC, angle(rad)):
1 1 10 3 5 9 0.87

Magnitude of cross product (AXB) = -7.000000
Dot product (A.B) = 13.000000
Length of A = 1.414214
Angle of vectors (A,B) = 0.493941
New vector (after rotating A) = (-0.119502, 1.409155)
Intersection point (BA and BC) = (10.000000, 3.000000)
```

3. Morleys Theorem

Morleys theorem states that that the lines trisecting the angles of an arbitrary plane triangle meet at the vertices of an equilateral triangle. For example in the figure below the tri-sectors of angles A, B and C has intersected and created an equilateral triangle DEF. Of course the theorem has various generalizations, in particular if all of the trisectors are intersected one obtains four other equilateral triangles. But in the original theorem only tri-sectors nearest to BC are allowed to intersect to get point D, tri-sectors nearest to CA are allowed to intersect point E and tri-sectors nearest to AB are intersected to get point F. Trisector like BD and CE are not allowed to intersect. So ultimately we get only one equilateral triangle DEF. Now your task is to find the Cartesian coordinates of D, E and F given the coordinates of A, B, and C. Please write a function void getIntsPoint(……) which accepts eight parameters, the former six parameters for three points of triangle, the latter two for returning coordinate of intersection point. For example, getIntsPoint(XA,YA, XB,YB, XC,YC, …) can obtain coordinate of intersection point D. You should use the functions in Problem 2.



Input

First line of the input file contains an integer N (0 < N < 5001) which denotes the number of test cases to follow. Each of the next lines contain six integers XA, YA, XB, YB, XC , YC . This six integers actually indicates that the Cartesian coordinates of point A, B and C are (XA,YA), (XB,YB) and (XC,YC) respectively. You can assume that the area of triangle ABC is not equal to zero, 0 ≤ XA, YA, XB, YB, XC , YC ≤ 1000 and the points A, B and C are in counter clockwise order.

Output

For each line of input you should produce one line of output. This line contains six floating point numbers XD, YD, XE, YE, XF, YF separated by a single space. These six floating-point actually means that the Cartesian coordinates of D, E and F are (XD,YD), (XE,YE), (XF,YF) respectively. Errors less than 10e−5 will be accepted.

Sample Input

```
2

1 1 2 2 1 2

0 0 100 0 50 50
```

Sample Output

```
1.316987 1.816987 1.183013 1.683013 1.366025 1.633975

56.698730 25.000000 43.301270 25.000000 50.000000 13.397460
```