Having a point, p(a, b)

To get the *angle* between this vector, We can use *arg* or *atan2* functions.
Those two function output the angle with radian, if we need t convert it into degrees, just *(Val* 180 / M_PI)*

```
#define ld long double
typedef ld T;
typedef complex<T> pt;

void RIM() {
ld a, b; cin >> a >> b;
pt p = {a, b};
cout << arg(p) * 180 / M_PI << '\n';
cout << atan2(b, a) << '\n';
}
```

Some Helpful functions: sq, abs, hypot, sgn

1. `T sq(pt p){`
   `return p.x * p.x + p.y * p.y;`
   `}`
2. `abs(p)` with `complex`
   If `p = {x, y}` is a `complex<T>` point, then:
   `abs(p) = sqrt(x*x + y*y)`
   It gives the distance from (0, 0) to (x, y) — the length of the vector.
   `abs` here equivalent to `hypot(a, b)`
3. `int sgn(T val){` --> *return if the Val Pos, Neg, or Zero. (With doubles)*
   `if(val > EPS) return 1;`
   `if(val <-EPS) return -1;`
   `else return 0;`
   `}`

Basic Trigonometry in Triangles (for CP)

Right Triangle with sides:

- `a` – adjacent (base)
- `b` – opposite (height)
- `r` – hypotenuse (i.e., `r = sqrt(a² + b²)`)

Trig Ratios:

```
sin(θ) = b / r
cos(θ) = a / r
tan(θ) = b / a
```

Law of Sines (for general triangles):

```
a / sin(α) = b / sin(β) = c / sin(γ) = 2R
```

Where `R` is the circumradius of the triangle. "نصف قطر المثلث في دائرة"

Law of Cosines

- In any triangle with sides a, b, and c, and angle θ between a and b:
  $cos(\theta) = a^2 + b^2 - c^2/2 * a * b$
- Use this to compute angles when all three sides are known.
- Rearranged to find side c if angle θ is known:
  $c^2 = a^2 + b^2 - 2ab \cdot cos(\theta)$

Tips:

- Use `atan2(b, a)` for angle from x-axis.
- Use `hypot(a, b)` instead of `sqrt(a*a + b*b)` for numerical stability.
- Memorize `sin²θ + cos²θ = 1` — useful for checks.

Transformations

1. Translation
2. Scaling
3. Rotating

```
pt translate(pt p, pt v){ // translate point p with vector v
return p+v;
}
```

```
pt scale(pt c, ld factor, pt p){ // scale point p with specific factor around point c
return c + (p-c) * factor;
}
```

```
pt rotate(pt p, pt c, ld a){ // To rotate point p around point c with a specific angle with radian
pt v = p - c; // vector between two points
pt rotate = {cos(a), sin(a)};
return c + rotate * v;
}
```

```
pt linearTransformation(pt p, pt q, pt r, pt fp, pt fq){
return fp + (r-p) * (fq-fp) / (q-p);
}
```

Dot product and cross product + angles

Dot Product

```
T dot(pt v, pt w){
return v.x * w.x + v.y * w.y;
}
```

Cross product

```
T cross(pt v, pt w){
return v.x * w.y - v.y * w.x;
}
```

to get angle between two vectors out from the origin point (0, 0)

```
T angle(pt v, pt w){
return acos(clamp(dot(v, w) / abs(v) / abs(w), (T)-1.0, (T)1.0));
}
```

to get an angle between a vector and x-axis (Polar angle)

```
const ld EPS = 1e-9;
const ld PI = 3.141592653589793238L;

int sgn(T val){
if(val > EPS) return 1;
if(val < -EPS) return -1;
else return 0;
}

void RIM() {
ld x, y;
cin >> x >> y;
ld c = atan2(y, x);
if(sgn(c) == -1)
```

```
c += 2 * PI;
cout << fixed << setprecision(10) << c;
}
```

To check if an angle between two vectors is perpendicular

`bool isPerp(pt v, pt w){` return fabs(dot(v, w)) < EPS; `}`

if we have a vector (a, b), vector(-b, a) will be perpendicular on it on it's left side.

```
pt perp(pt v){
return (-v.y, v.x);
}
```