```cpp
/*
    | Point Class |
    Desc: A generic point class with some helper funcs
    Source: KawakiMeido
    State: Untested and VERY buggy lmao
*/

#define X real()
#define Y imag()

template <typename T>
class Point {
public:
    static constexpr double EPS = 1e-6;

    std::complex<T> p;

    // Contructors
    Point(T x = 0, T y = 0) : p(x, y) {}
    explicit Point(const std::complex<T>& val) : p(val) {}

    // Accessors
//    T real() { return p.real(); }
    T real() const { return p.real(); }
//    T imag() { return p.imag(); }
    T imag() const { return p.imag(); }

    void setX(int x) {
        p.real(x);
    }

    void setY(int y) {
        p.imag(y);
    }

    // Comparisons
    bool operator==(const Point& other) const {
        if constexpr (std::is_floating_point_v<T>) {
            return (std::abs(p.real() - other.p.real()) < EPS) &&
                   (std::abs(p.imag() - other.p.imag()) < EPS);
        } else {
            return p == other.p;
        }
    }
    bool operator≠(const Point& other) const { return !(*this == other); }

    // Arithmetics
    Point& operator+=(const Point& other) { p += other.p; return *this; }
    Point& operator-=(const Point& other) { p -= other.p; return *this; }
```

```
50      friend Point operator+(Point a, const Point& b) { a += b; return a; }
51      friend Point operator-(Point a, const Point& b) { a -= b; return a; }
52
53      // Helper Functions
54      friend T dot(const Point& a, const Point& b)   { return (std::conj(a.p) *
    b.p).real(); }
55      friend T cross(const Point& a, const Point& b) { return (std::conj(a.p) *
    b.p).imag(); }
56      friend T sqdist(const Point& a, const Point& b){ return std::norm(a.p -
    b.p); }
57      friend T dist(const Point& a, const Point& b)  { return std::abs(a.p - b.p);
    }
58      friend long double angle(const Point& a, const Point& b) { return
    std::arg(b.p - a.p); }
59      friend long double slope(const Point& a, const Point& b) { return
    std::tan(std::arg(a.p - b.p)); }
60  };
```