Q-18

(a) Two points P_1 and P_2 are ~~equal.~~ ~~equivalent.~~ equal

i.e $P_1 \sim P_2$ if and only if $P_1.x = P_2.x$, $P_1.y = P_2.y$, $inf_1 = inf_2$ & $nac_1 = nac_2$

i.e. $(x_1, y_1, inf_1, nac_1) \sim (x_2, y_2, inf_2, nac_2) \iff$

$$x_1 = x_2,\ y_1 = y_2,\ inf_1 = inf_2,\ nac_1 = nac_2$$

where $x, y \in N$, $inf, nac \in \{0, 1\}$ ($\because$ {True, False})

Here point P is represented by tuple (x, y, inf, nac)

This description holds the properties of reflexive, symmetric and transitivity.

① Reflexive.

Let (a, b, inf, nac) be the tuple

then

$$a = a,\ b = b,\ inf = inf,\ nac = nac.$$

So. $(a, b, inf, nac) \sim (a, b, inf, nac)$

Symmetric.

② Let $\overset{P_1}{(a_1, b_1, inf_1, nac_1)}$ & $\overset{P_2}{(a_2, b_2, inf_2, nac_2)}$ be two points

~~Now we know~~

~~$P_1 \sim P_2$~~

Suppose $P_1 \sim P_2$ i.e $a_1 = a_2$, $b_1 = b_2$, $inf_1 = inf_2$, $nac_1 = nac_2$

$\equiv$ $a_2 = a_1$, $b_2 = b_1$, $inf_2 = inf_1$, $nac_2 = nac_1$

$\equiv (a_2, b_2, inf_2, nac_2) \sim (a_1, b_1, inf_1, nac_1)$

$\equiv P_2 \sim P_1$

Transitivity

Let P_1, P_2, P_3 be Points

Now Supposes $P_1 \sim P_2$ & $P_2 \sim P_3$

i.e $(a_1, b_1, inf_1, nac_1) \sim (a_2, b_2, inf_2, nac_2)$

$a_1 = a_2,\ b_1 = b_2,\ inf_1 = inf_2,\ nac_1 = nac_2$ — (1)

$(a_2, b_2, inf_2, nac_2) \sim (a_3, b_3, inf_3, nac_3)$

$a_2 = a_3,\ b_2 = b_3,\ inf_2 = inf_3,\ nac_2 = nac_3$ — (2)

from (1) & (2)

$a_1 = a_3,\ b_1 = b_3,\ inf_1 = inf_3,\ nac_1 = nac_3$

thus $P_1 \sim P_3$

(b)
```
bool eq(const Point &other) {
    if (isInf() == other.isInf() && isPosi() == other.isPosi())
        return True.
else
```

(b) bool eq(const Point &other){

```
if (isInf()){
    if (other.isInf() != true)
        return false;

    if (isPosi() == other.isPosi())
        return true.

    else return false.
}

if (isNac()){
    if (isNac() == other.isNac())
        return true
    else return false
}

if (x == other.getX() && y == other.getY())
    return true.

else return false.
```

}.

(c) The weak ordering on Point which is of less than if defined then it is ~~equivalent~~ anti symmetry, anti reflexive and transitive.

(d) ~~bool lt(const Point& other){~~

~~if (isNaC()){~~

~~if (other.isNaC() == true)~~
~~return false~~

(d)
```
bool lt(const Point & other){
    if (x < other.getx())
        return true
    if (x > other.getx())
        return false
    if (x == other.getx() && y < other.gety())
        return true;
    else return false
}.
```

(e) Here if a point is Inf and has x == 1 we return Constant representing positive infinity. (Say const is INT_MAX)

if a point is Inf and has x == -1 we return Constant2 representing negative infinity. (Say const is INT_MIN)

if a point in NaC then we return Constant 3 representing not a coordinate. We map it to say 0.

for normal point we multiple x cordinate with some large prime number and add y coordinate to it.

```
unsigned hash() {

    if(isInf() && getx() == 1)
        return INT_MAX.

    if (isInf() && getx() == -1)
        return INT_MIN

    if (isNaC())
        return 0;

    unsigned hashV = 0

    hashV = 33739 x unsigned(x) + unsigned(y)

    return hashV;

}
```

Here 33739 is one prime number.

The above hash function defined hash following properties

1) Computation is $\Theta(1)$

2) It is deterministic → Always return integer which is same for each point on every call.

3) If two points P_1 & P_2 are equal then the hash calculated is also equal.

if 2 positive infinity taken then INT_MAX is returned for both.