Eight Way Symmetry

Midpoint Line Drawing Algorithm

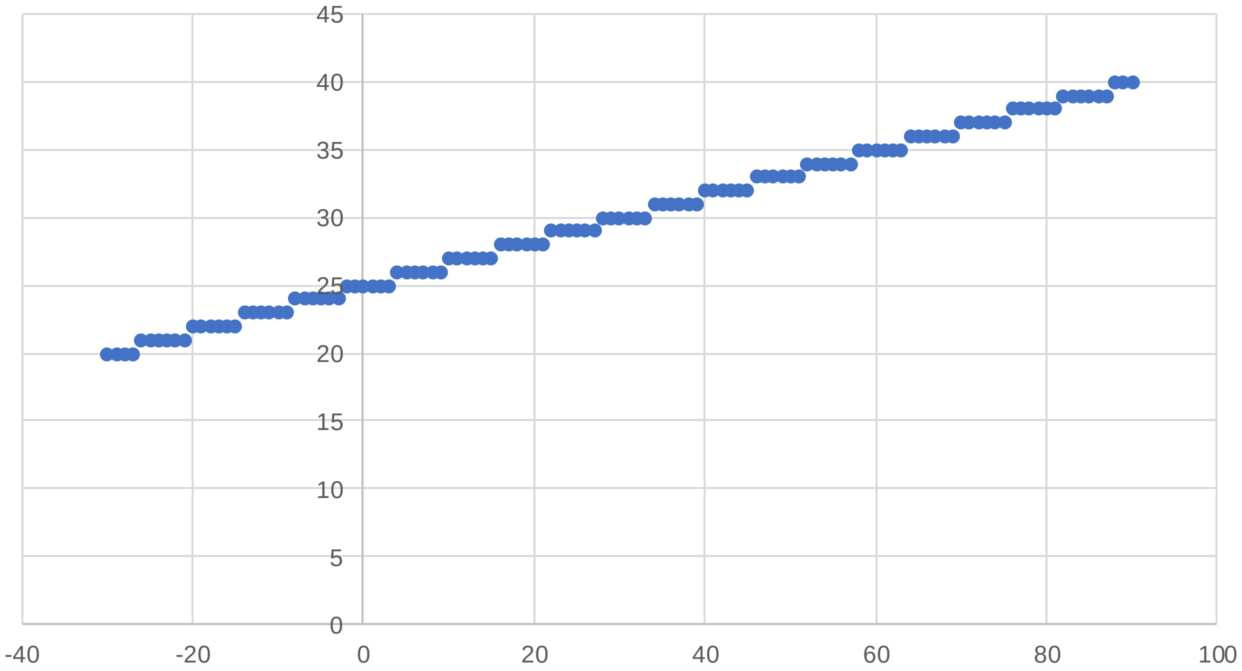
Midpoint Line Drawing Algorithm

```
Midpoint ($x_1$, $y_1$, $x_2$, $y_2$){
    dx= $x_2 - x_1$ ; dy = $y_2 - y_1$ ;
    D = 2*dy – dx; $\Delta$NE = 2*(dy-dx) ; $\Delta$E = 2*dy ;
    x= $x_1$ ; y = $y_1$ ;
    while( $x \leq x_2$){
        Draw(x, y);
        x++ ;
        if (D>0){
            y++;
            D = D+ $\Delta$NE ;
        }
        else{
            D = D+ $\Delta$E ;
        }
    }
}
```

(-30, 20) to (90, 40)

dx= 90 + 30 = 120 ; dy = 40- 20 = 20;

D = 2*20 – 120 = -80; ∆NE = 2*(20-120) = -200 ; ∆E = 2*20 = 40;



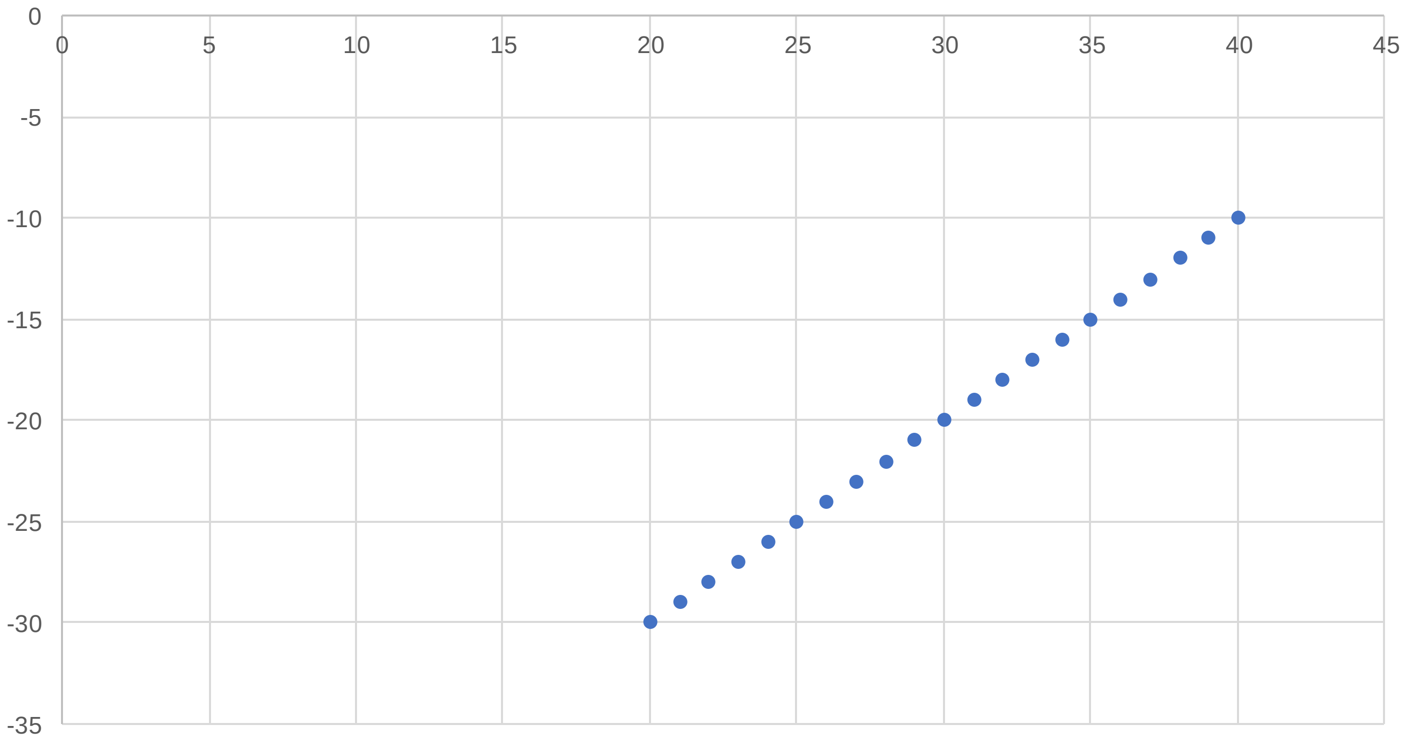
$$m = \frac{20}{120} = 0.167 < 1$$

X	Y	D
-30	20	-80
-29	20	-40
-28	20	0
-27	20	40
-26	21	-160
-25	21	-120
-24	21	-80
-23	21	-40
-22	21	0
-21	21	40
-20	22	-160
-19	22	-120
-18	22	-80
-17	22	-40
-16	22	0
-15	22	40
-14	23	-160
-13	23	-120
-12	23	-80
-11	23	-40

(20, -30) to (40, 90)

dy= 90 + 30 = 120 ; dx = 40- 20 = 20;

D = 2*120 – 20 = 220; ∆NE = 2*(120-20) = 200 ; ∆E = 2*120 = 240;

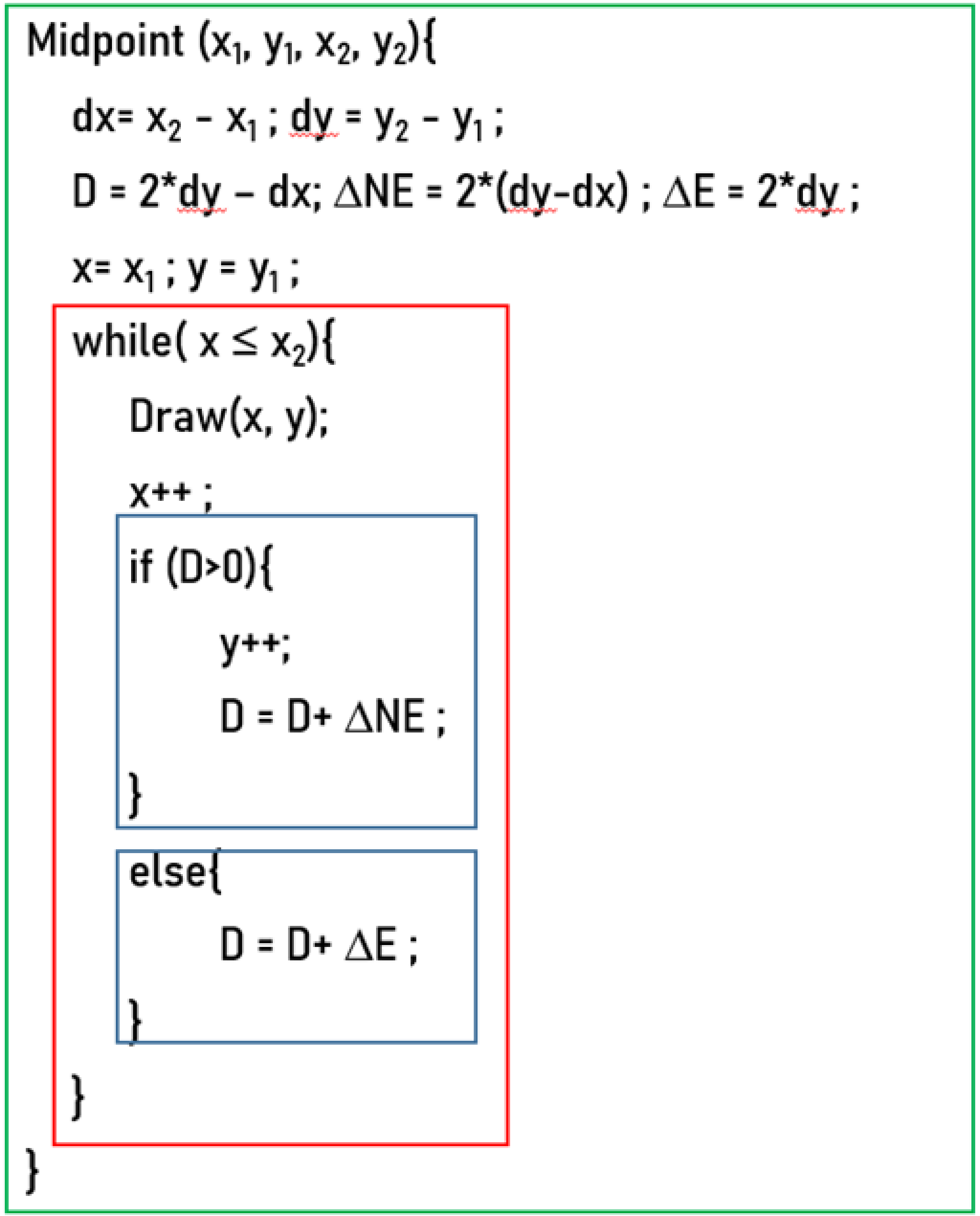


$$m = \frac{120}{20} = 6 > 1$$

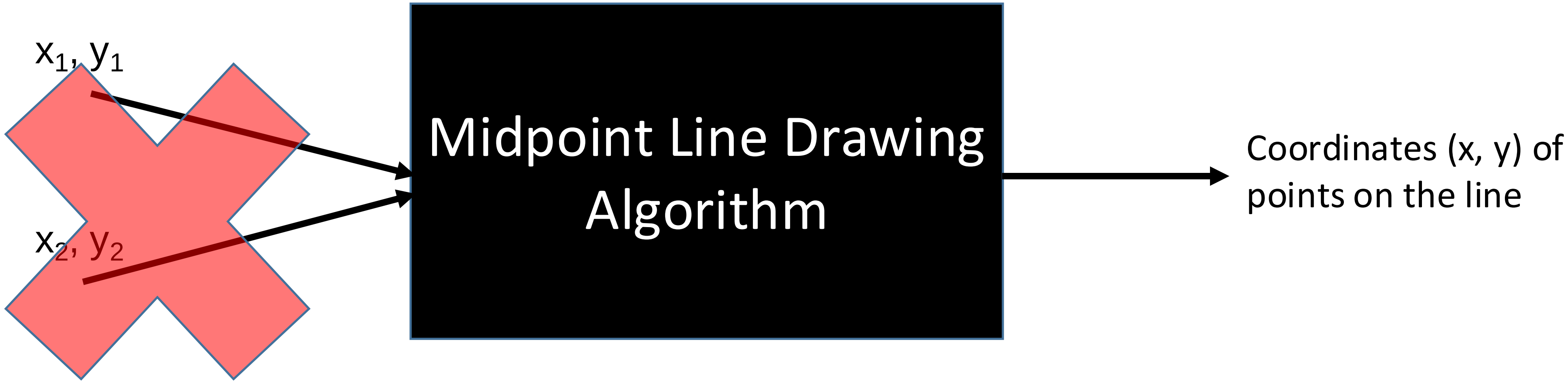
X	Y	D
20	-30	220
21	-29	420
22	-28	620
23	-27	820
24	-26	1020
25	-25	1220
26	-24	1420
27	-23	1620
28	-22	1820
29	-21	2020
30	-20	2220
31	-19	2420
32	-18	2620
33	-17	2820
34	-16	3020
35	-15	3220
36	-14	3420
37	-13	3620
38	-12	3820

(30, -20) to (-90, 40)

-
- If we start from (30, -20), then we need to decrement x to reach (-90, 40)
- If we start from (-90, 40), x will be incremented to reach (30, -20) but y needs to be decremented!
- m = $\frac{60}{-120}$ = - 0.5 < 0



```
Midpoint ($x_1$, $y_1$, $x_2$, $y_2$){
    dx= $x_2$ - $x_1$ ; dy = $y_2$ - $y_1$ ;
    D = 2*dy – dx; ΔNE = 2*(dy-dx) ; ΔE = 2*dy ;
    x= $x_1$ ; y = $y_1$ ;
    while( x ≤ $x_2$){
        Draw(x, y);
        x++ ;
        if (D>0){
            y++;
            D = D+ ΔNE ;
        }
        else{
            D = D+ ΔE ;
        }
    }
}
```



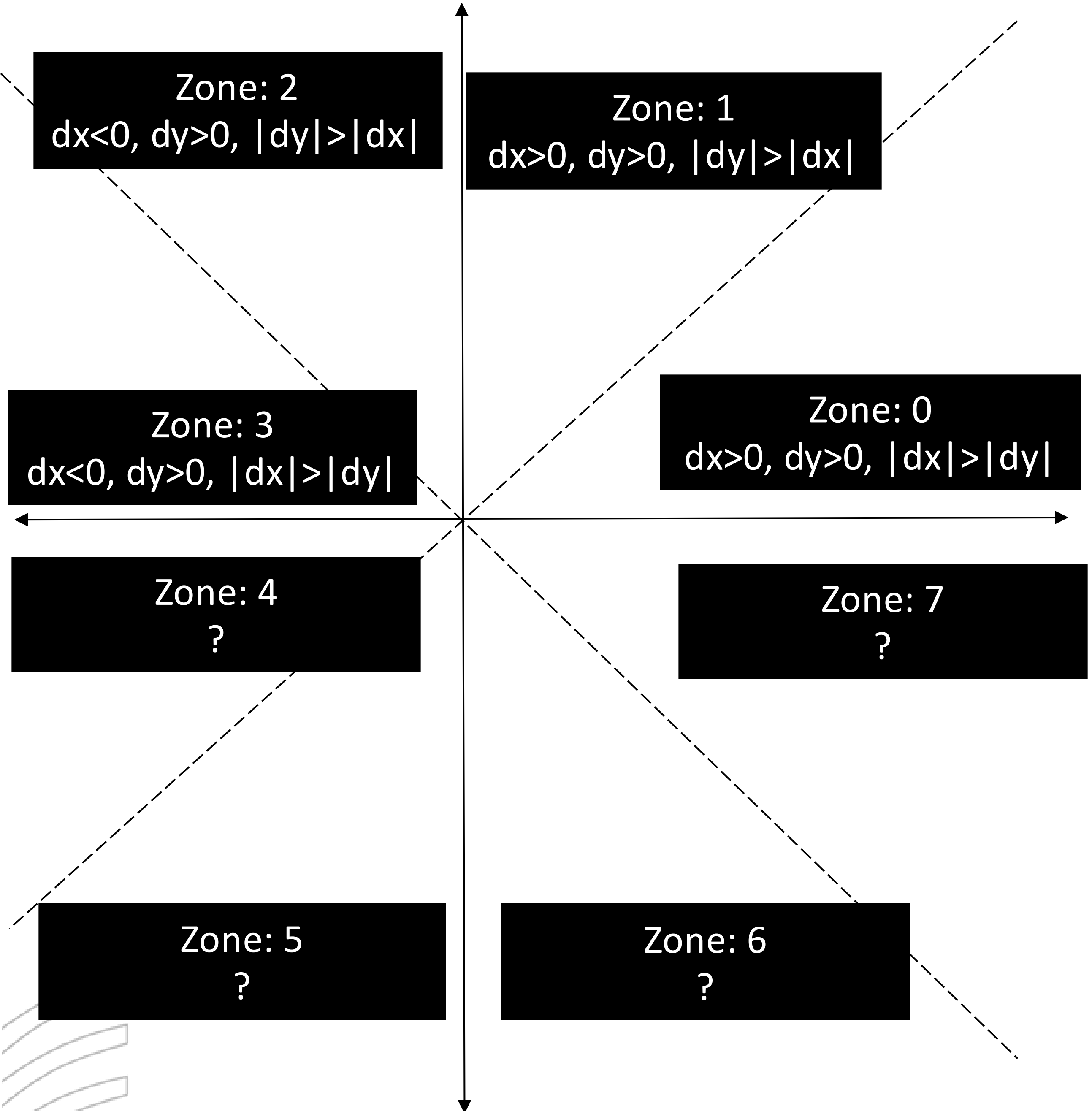
x_1, y_1

x_2, y_2

Midpoint Line Drawing Algorithm

Coordinates (x, y) of points on the line

Eight Way Symmetry



Zone: 2
dx<0, dy>0, |dy|>|dx|

Zone: 1
dx>0, dy>0, |dy|>|dx|

Zone: 3
dx<0, dy>0, |dx|>|dy|

Zone: 0
dx>0, dy>0, |dx|>|dy|

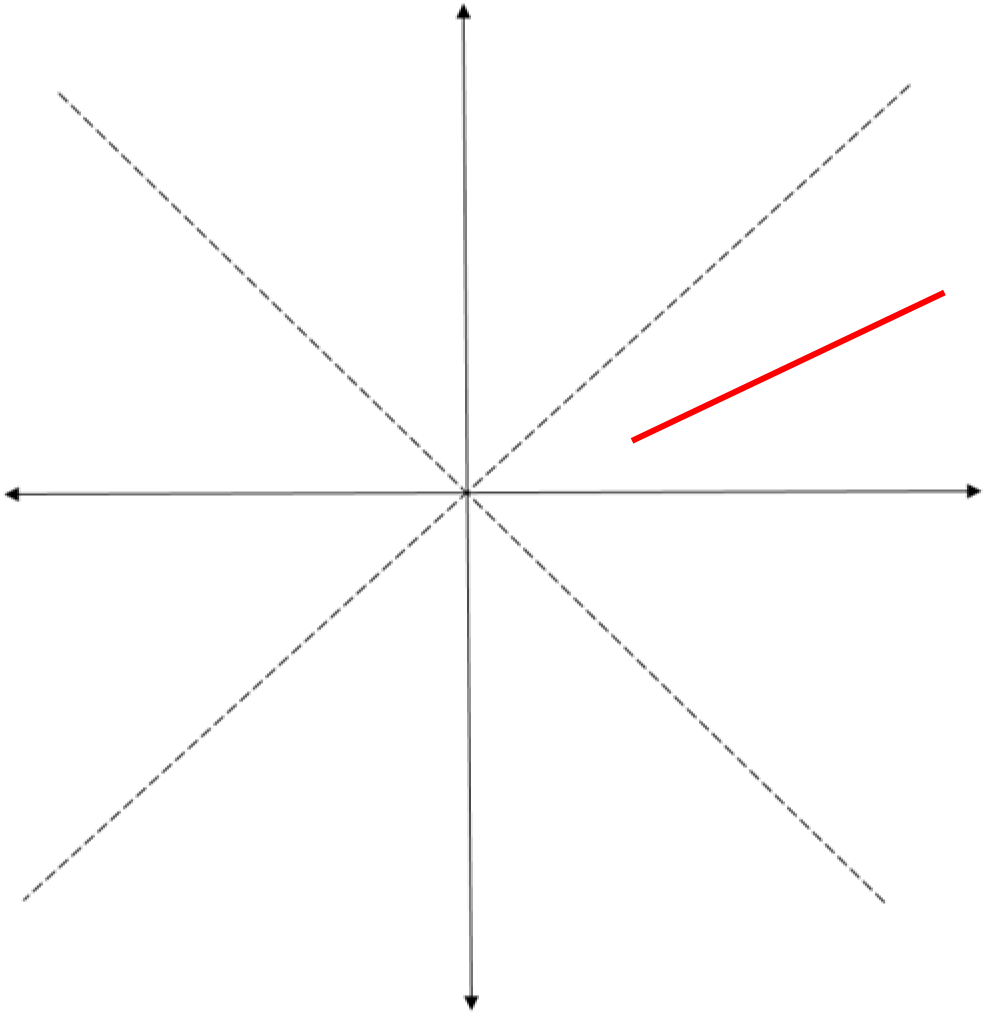
Zone: 4
?

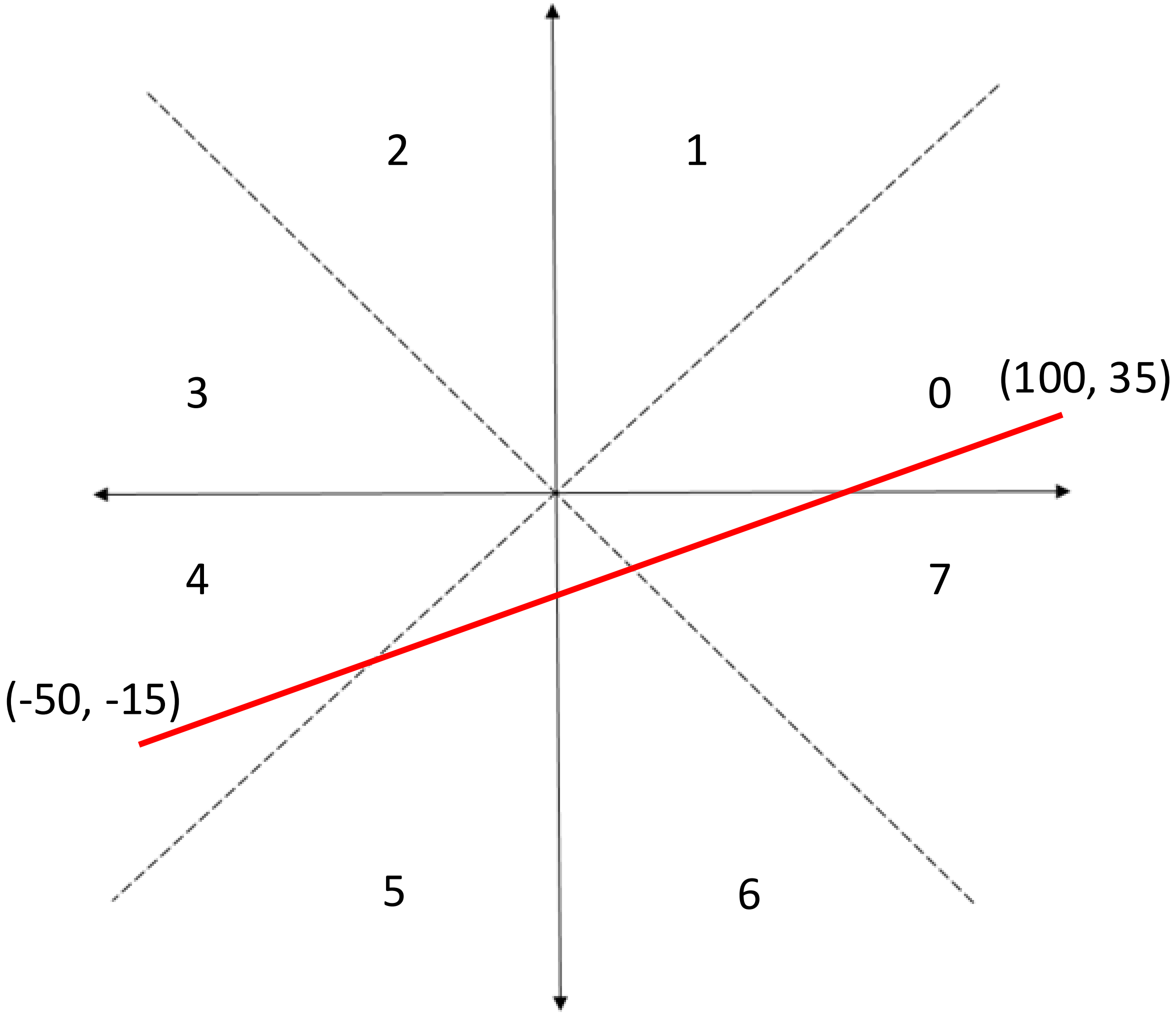
Zone: 7
?

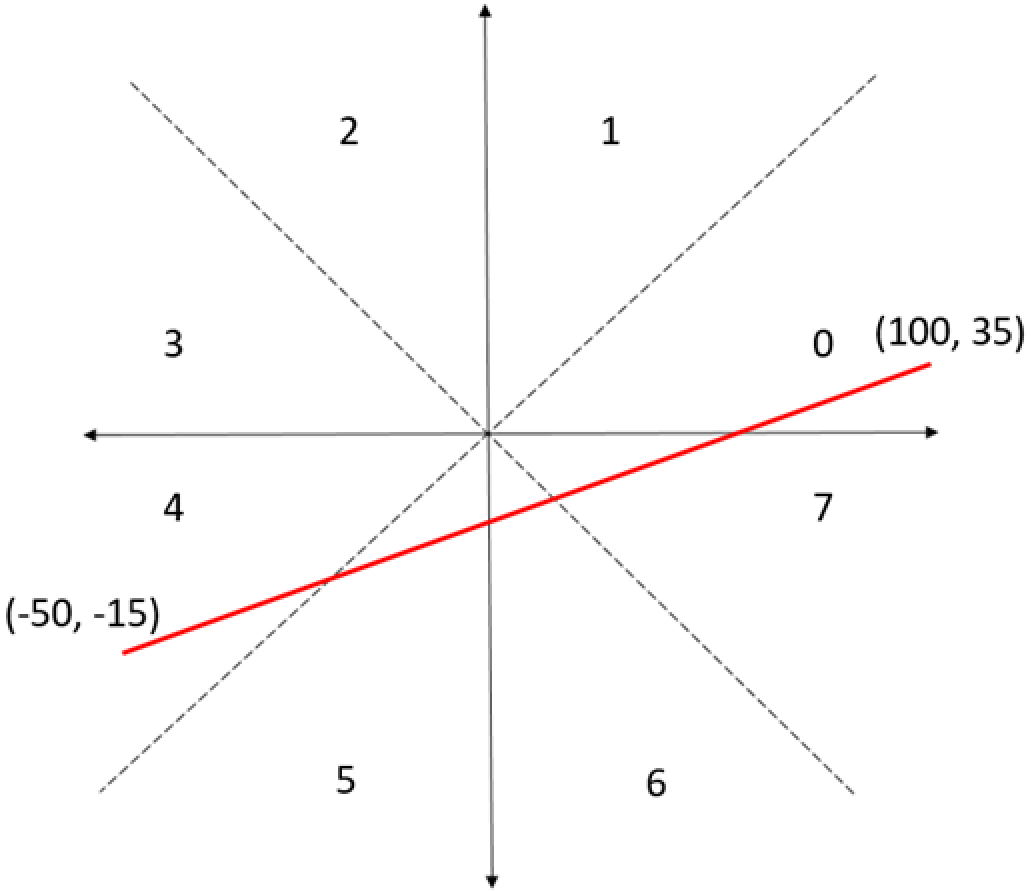
Zone: 5
?

Zone: 6
?

```
FindZone($x_1$, $y_1$, $x_2$, $y_2$){
   dx= $x_2$ - $x_1$ ; dy = $y_2$ - $y_1$ ;

   if(|dx| > |dy|){
      if(dx>0 && dy>0) zone = 0;
      else if(dx<0 && dy>0) zone =3;
      else if (? ?) zone = ? ;
      else if (? ?) zone = ?
   }

   else{
      if(dx>0 && dy>0) zone = 1;
      else if(dx<0 && dy>0) zone =2;
      else if (? ?) zone = ? ;
      else if (? ?) zone = ?
   }

}
```




2
1
3
0
(100, 35)
4
7
(-50, -15)
5
6



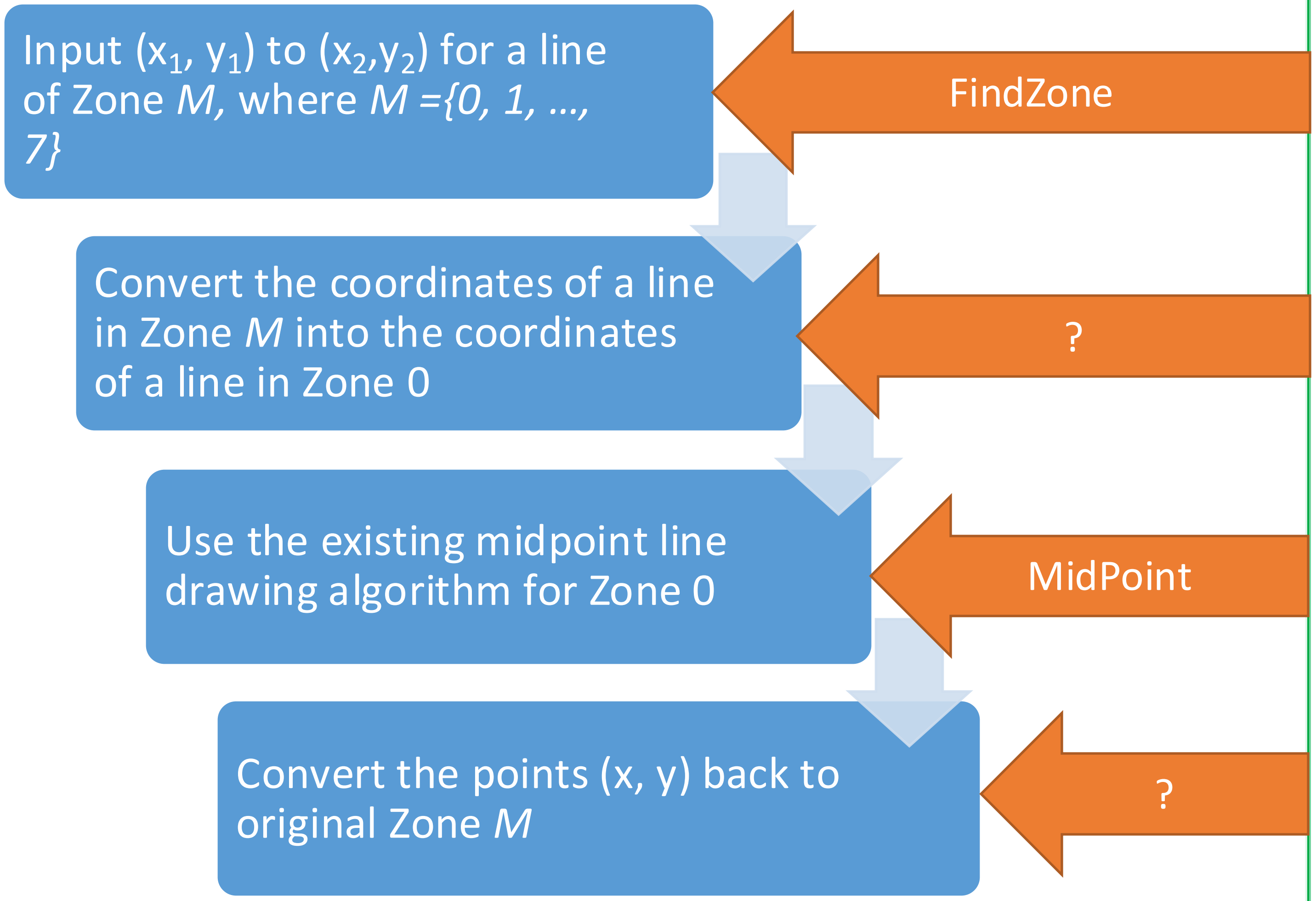
$dx = 100 + 50 = 150 > 0$

$dy = 35 + 15 = 50 > 0$

$|dx| > |dy|$

Zone = 0

How do we utilize the zones?

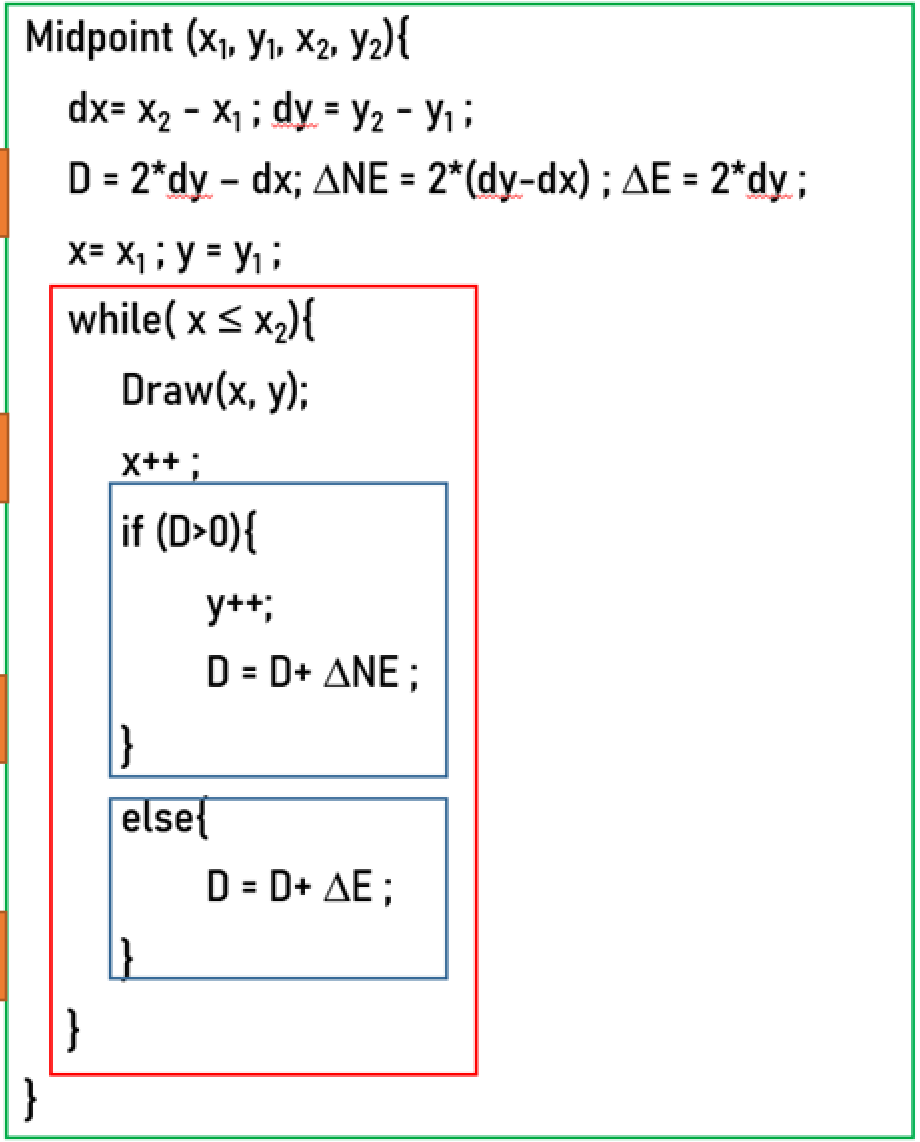


Input (x_1, y_1) to (x_2,y_2) for a line of Zone *M, where M ={0, 1, …, 7}*

Convert the coordinates of a line in Zone *M* into the coordinates of a line in Zone 0

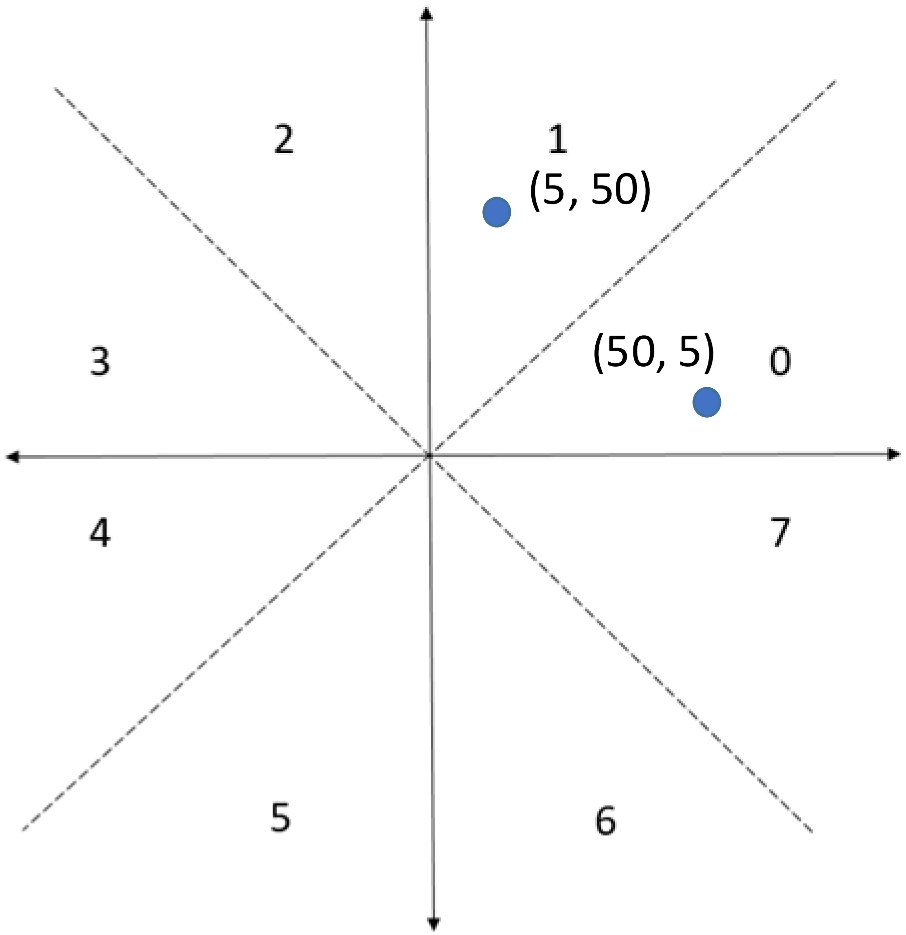
Use the existing midpoint line drawing algorithm for Zone 0

Convert the points (x, y) back to original Zone *M*



```
Midpoint ($x_1$, $y_1$, $x_2$, $y_2$){
    dx= $x_2 - x_1$ ; dy = $y_2 - y_1$ ;
    D = 2*dy – dx; ΔNE = 2*(dy-dx) ; ΔE = 2*dy ;
    x= $x_1$ ; y = $y_1$ ;
    while( $x \le x_2$){
        Draw(x, y);
        x++ ;
        if (D>0){
            y++;
            D = D+ ΔNE ;
        }
        else{
            D = D+ ΔE ;
        }
    }
}
```

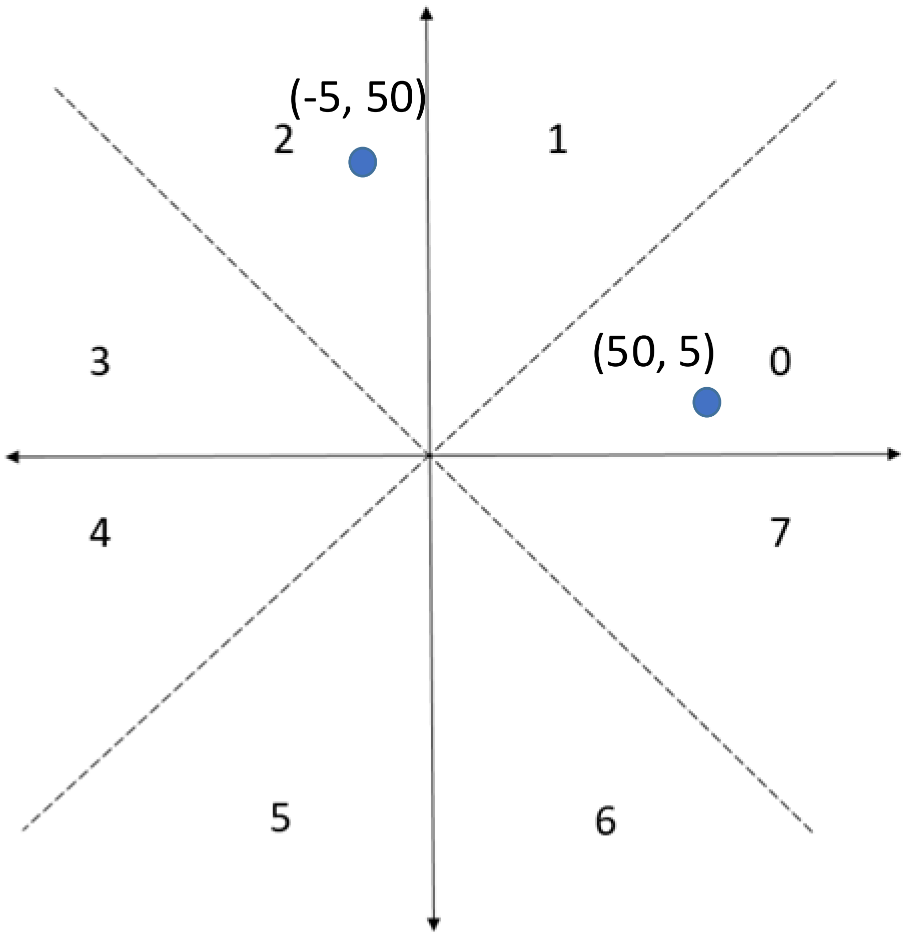
Convert the coordinates of Zone *M* into the coordinates of Zone 0
Zone 1 → Zone 0



Coordinates in Zone 1: (X, Y) becomes (Y , X) in Zone 0

```
ConvertToZone0 (X, Y, zone){

    if (zone == 1){
       x = Y, y = X
    }
    return (x, y)

}
```

Convert the coordinates of Zone *M* into the coordinates of Zone 0
Zone 2 → Zone 0

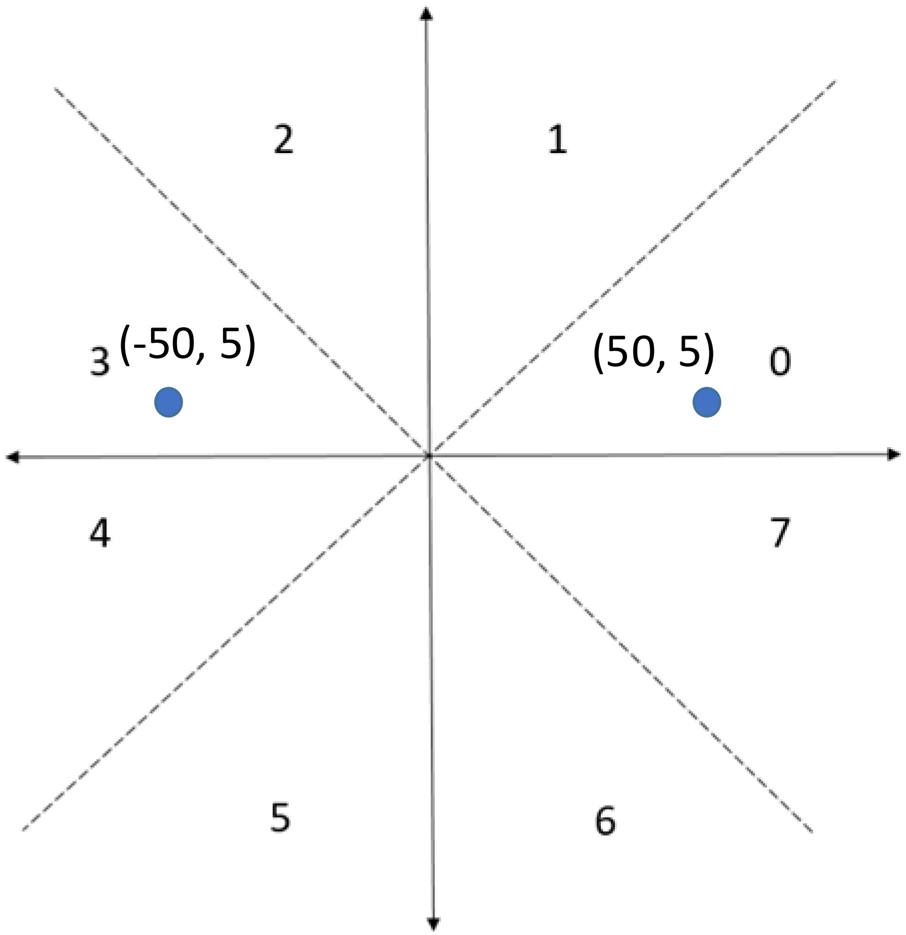


Coordinates in Zone 2: (X, Y) becomes (Y , - X) in Zone 0

```
ConvertToZone0 (X, Y, zone){

    if (zone == 1){
       x = Y, y = X
    }
    else if (zone ==2){
       x = Y, y = -X
    }
    return (x, y)

}
```

Convert the coordinates of Zone *M* into the coordinates of Zone 0

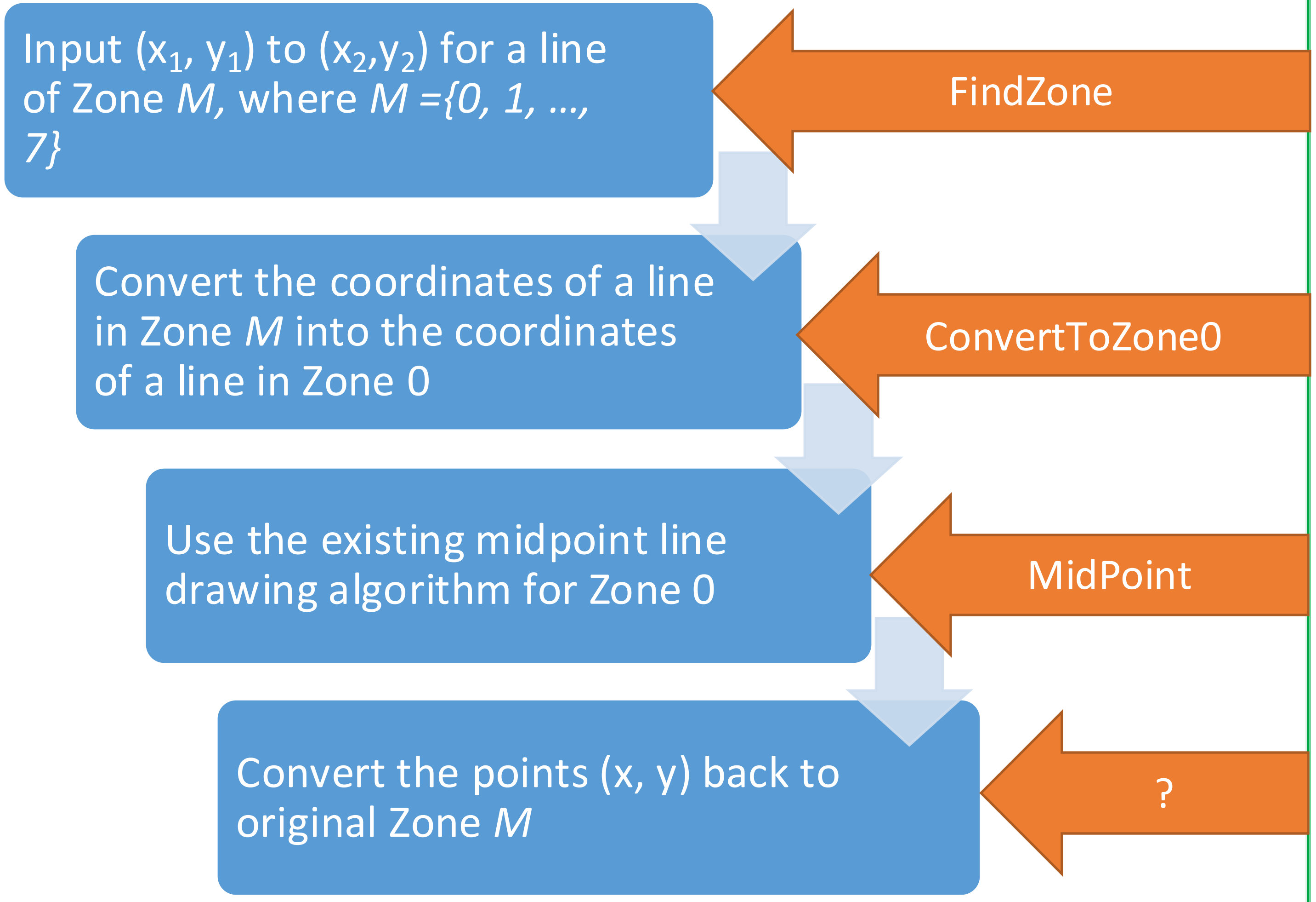
Zone 3 → Zone 0



Coordinates in Zone 3: (X, Y) becomes (-X, Y) in Zone 0

```
ConvertToZone0 (X, Y, zone){

    if (zone == 1){
      x = Y, y = X
    }
    else if (zone ==2){
      x = Y, y = -X
    }
    else if (zone ==3){
      x = -X , y = Y
    }
    ….
    return (x, y)

}
```

DIY for zone 4, 5, 6, 7

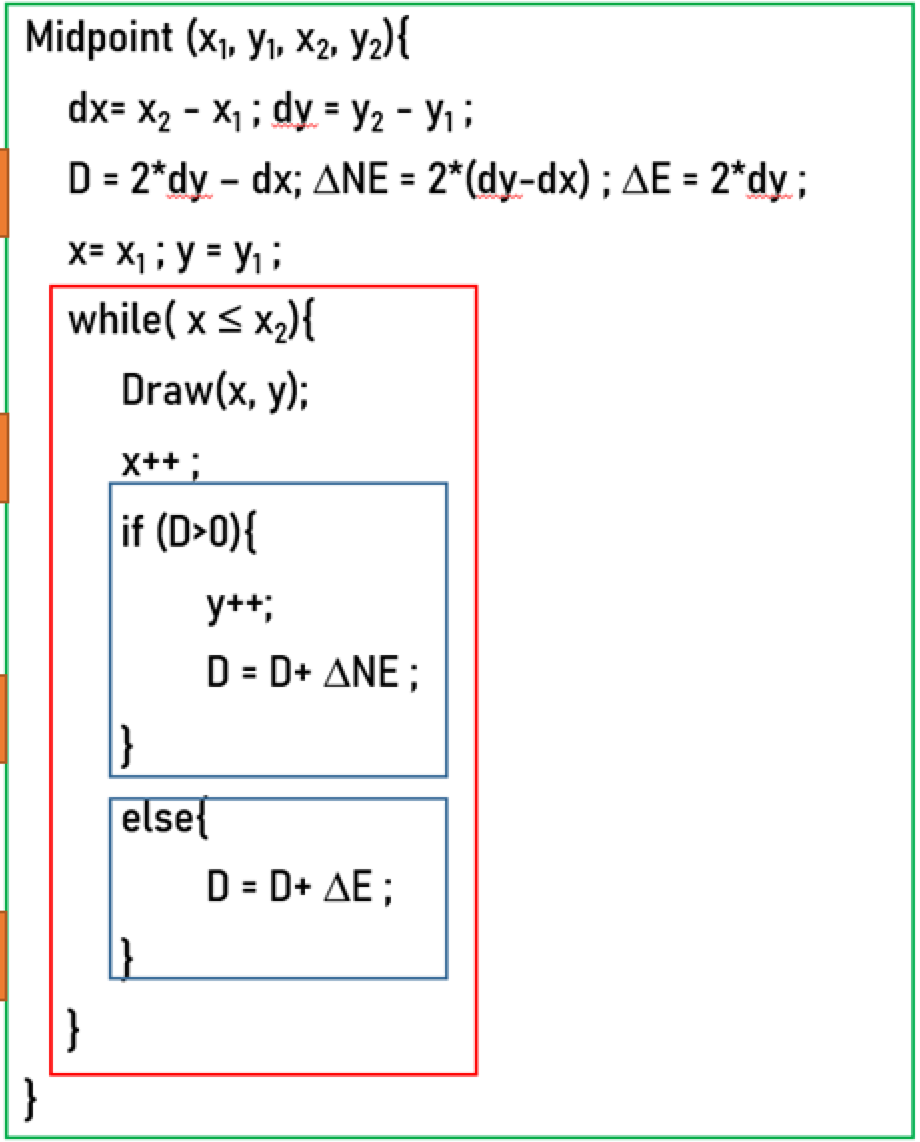


Input (x_1, y_1) to (x_2,y_2) for a line of Zone *M, where M ={0, 1, …, 7}*

Convert the coordinates of a line in Zone *M* into the coordinates of a line in Zone 0

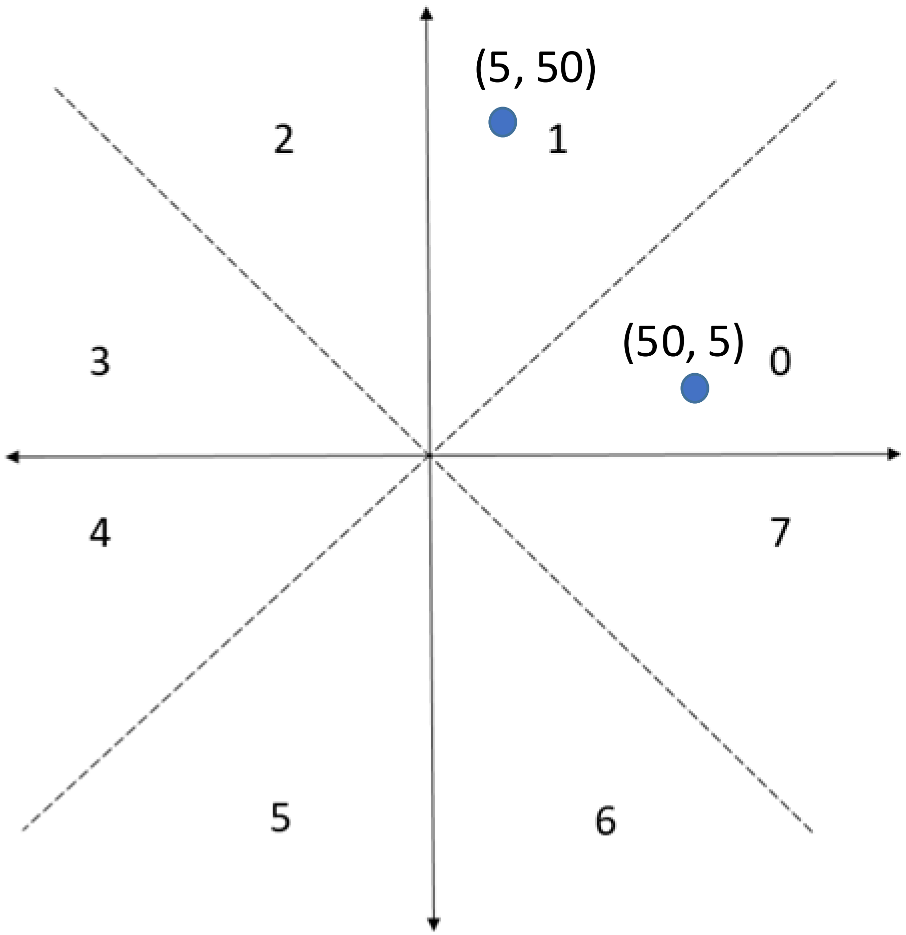
Use the existing midpoint line drawing algorithm for Zone 0

Convert the points (x, y) back to original Zone *M*



```
Midpoint ($x_1$, $y_1$, $x_2$, $y_2$){
    dx= $x_2 - x_1$ ; dy = $y_2 - y_1$ ;
    D = 2*dy – dx; $\Delta$NE = 2*(dy-dx) ; $\Delta$E = 2*dy ;
    x= $x_1$ ; y = $y_1$ ;
    while( $x \leq x_2$){
        Draw(x, y);
        x++ ;
        if (D>0){
            y++;
            D = D+ $\Delta$NE ;
        }
        else{
            D = D+ $\Delta$E ;
        }
    }
}
```

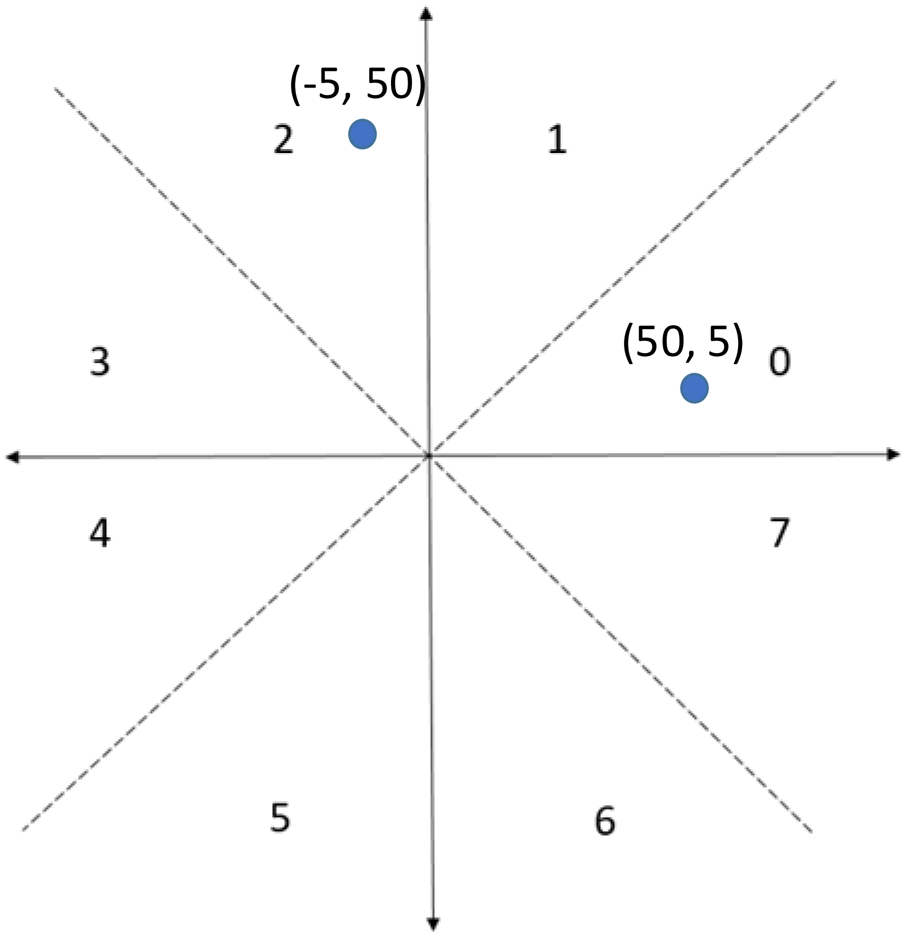
Go back to original zone M
Zone 0 → Zone 1



Coordinates in Zone 0: (X, Y) becomes (Y , X) in Zone 1

```
OriginalZone (X, Y, zone){

    if (zone == 1){
       x = Y, y = X
    }
    return (x, y)

}
```

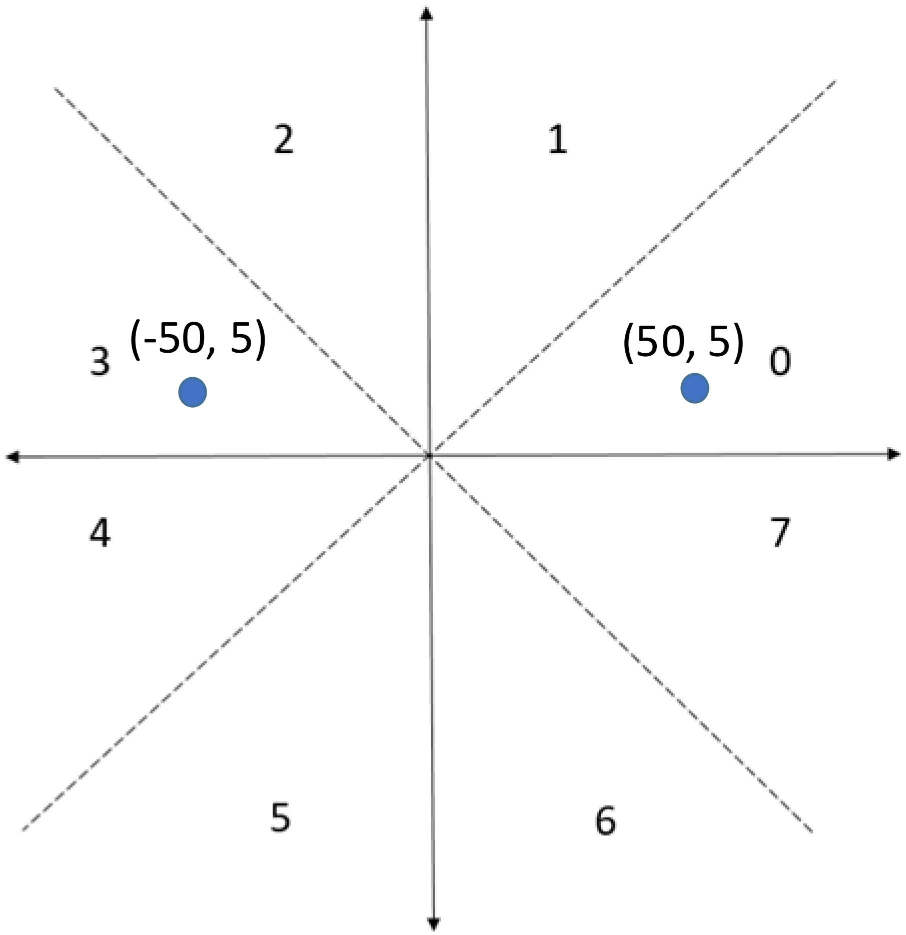
Go back to original zone M
Zone 0 → Zone 2



Coordinates in Zone 0: (X, Y) becomes (-Y , X) in Zone 2

```
OriginalZone (X, Y, zone){

    if (zone == 1){
       x = Y, y = X
    }
    else if(zone == 2){
       x = -Y, y = X
    }
    return (x, y)

}
```

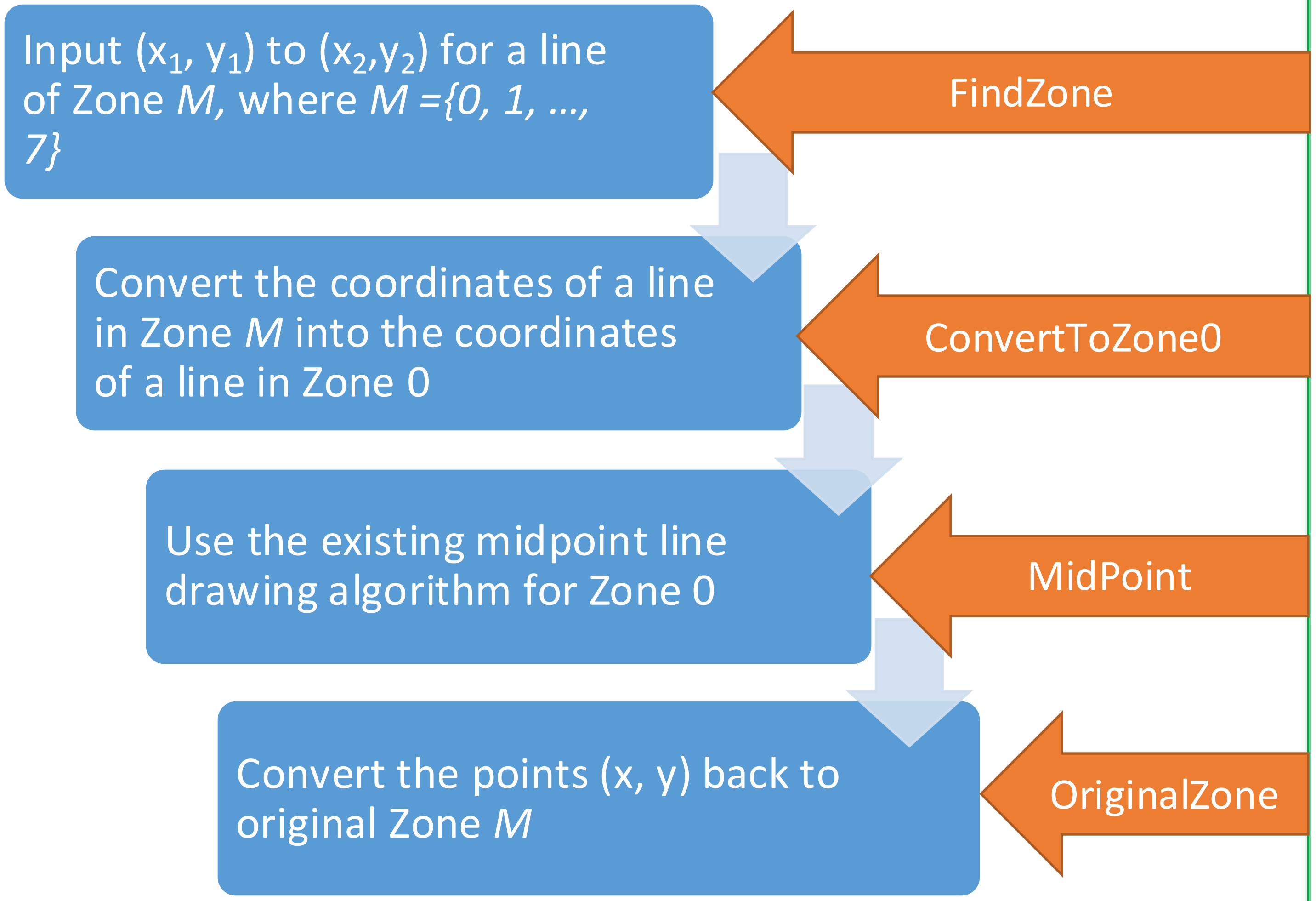
Go back to original zone M
Zone 0 → Zone 3



Coordinates in Zone 0: (X, Y) becomes (-X , Y) in Zone 3

```
OriginalZone (X, Y, zone){

    if (zone == 1){
       x = Y, y = X
    }
    else if(zone == 2){
       x = -Y, y = X
    }
   else if (zone ==3){
       x = -X , y = Y
    }
    ….
    return (x, y)

}
```

DIY for zone 4, 5, 6, 7

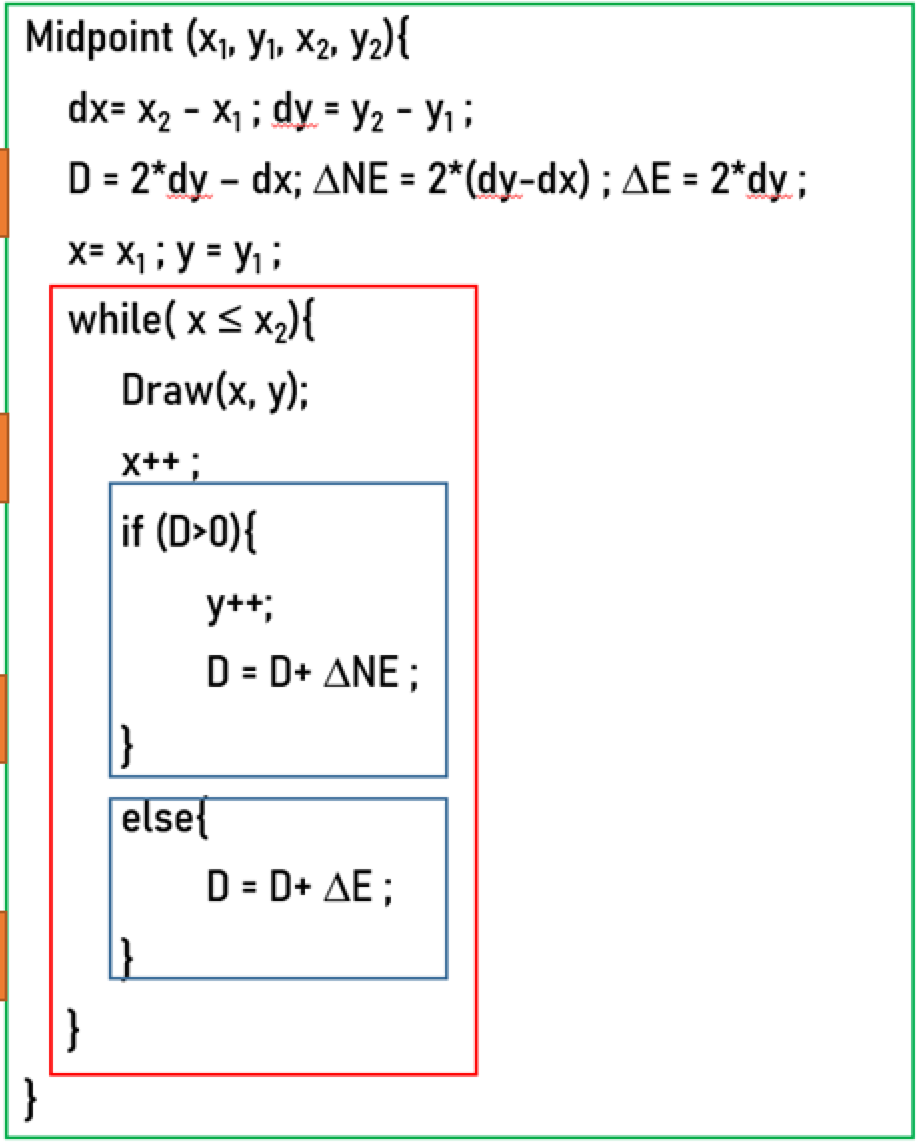


Input (x_1, y_1) to (x_2,y_2) for a line of Zone *M*, where *M ={0, 1, …, 7}*

Convert the coordinates of a line in Zone *M* into the coordinates of a line in Zone 0

Use the existing midpoint line drawing algorithm for Zone 0

Convert the points (x, y) back to original Zone *M*



```
Midpoint ($x_1$, $y_1$, $x_2$, $y_2$){
    $dx = x_2 - x_1$ ; $dy = y_2 - y_1$ ;
    $D = 2*dy - dx$; $\Delta NE = 2*(dy-dx)$ ; $\Delta E = 2*dy$ ;
    $x = x_1$ ; $y = y_1$ ;
    while( $x \le x_2$){
        Draw(x, y);
        x++ ;
        if (D>0){
            y++;
            $D = D + \Delta NE$ ;
        }
        else{
            $D = D + \Delta E$ ;
        }
    }
}
```

(-10,-20) to (-20, 70)

dx = -20 +10 = -10 < 0

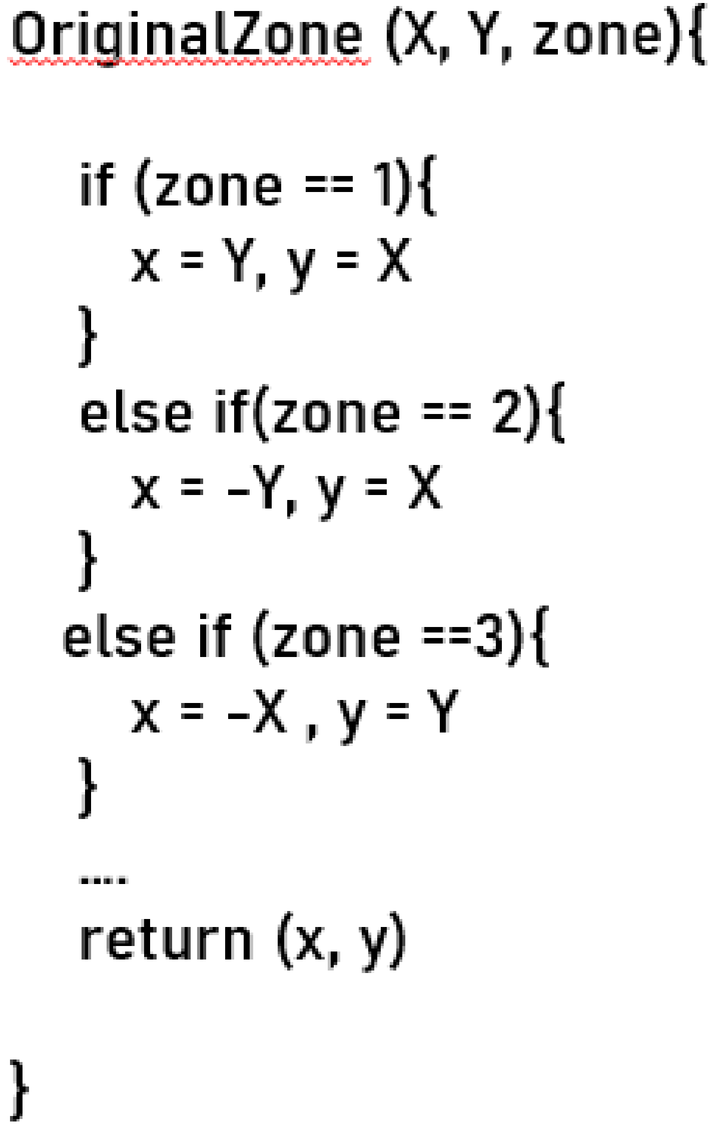
dy = 70 + 20 = 90 > 0

|dy| > |dx|, zone = 2

(-10, -20) → (-20, 10) and (-20, 70) →(70, 20)

dx’ = 70 + 20 = 90, dy’ = 20-10 = 10
D = 2*10-90 = -70, ΔNE = 2*(10-90)= -160, ΔE=2*10=20

X’	Y’	D	X	Y
-20	10	-70	-10	-20
-19	10	-50	-10	-19
-18	10	-30	-10	-18
-17	10	-10	-10	-17
-16	10	10	-10	-16
-15	11	-150	-11	-15
-14	11	-130	-11	-14



```
OriginalZone (X, Y, zone){

    if (zone == 1){
      x = Y, y = X
    }
    else if(zone == 2){
      x = -Y, y = X
    }
   else if (zone ==3){
      x = -X , y = Y
    }
    ....
    return (x, y)

}
```