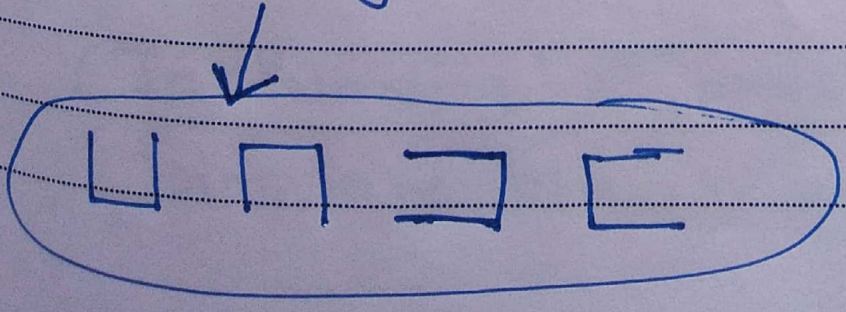
Hilbert Curve

- A Hilbert curve is a continuous space-filling curve.
- This is actually a fractal. If you closely observe, the pattern you see looks just the same as itself.

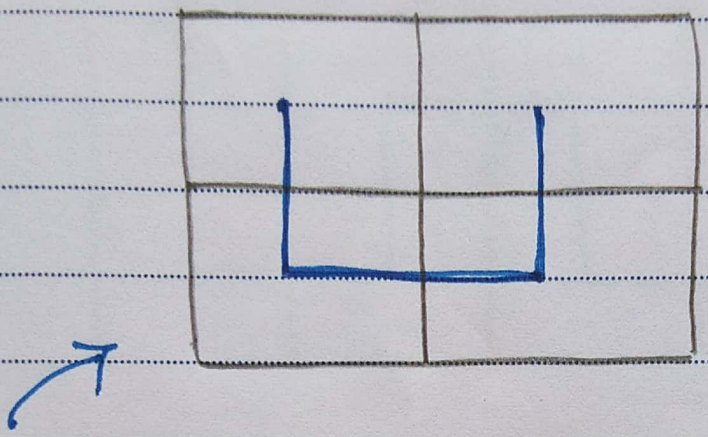
- The basic elements of the hilbert curves are:
  - → cups (a square with one side open) &
  - → joins (a vector that joins two cups)

- The open side of a cup can be top, bottom, left or right.



2026

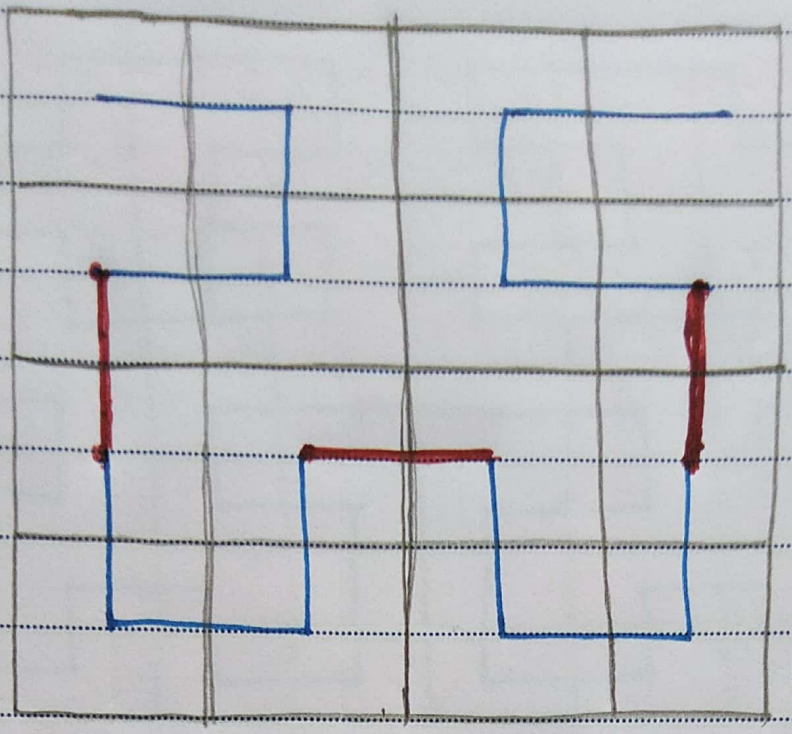
- A first order Hilbert curve is just a single cup.
- It fills a 2×2 space.



[First order Hilbert curve is shown by blue color & that 2×2 space is shown by pencil.]

- The second order Hilbert curve replaces that cup by four cups, which are linked together by three joins.

(In following 2nd order hilbert curve, joins are shown by red color)

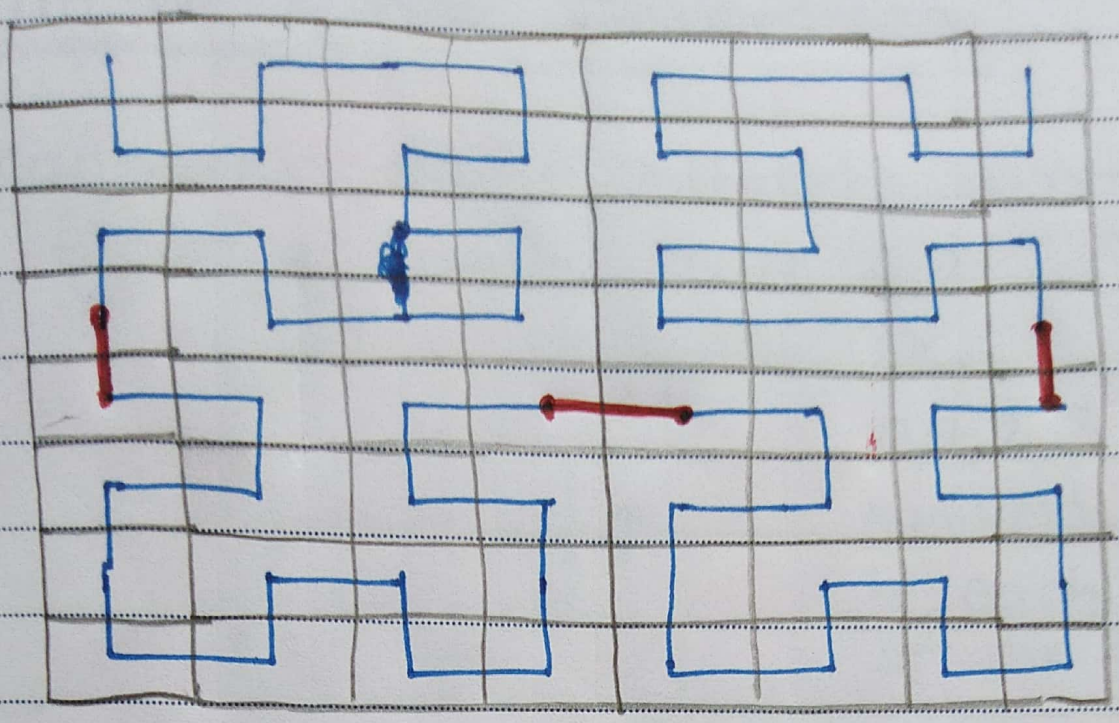


Here, we double the size of the grid to make a 4x4 grid.

- Each next order repeats the process.

  ↳ For 3rd order hilbert curve, we use 4 2nd order hilbert curves & 3 joins to join them.

  For 4th order hilbert curve, we use 4 3rd order hilbert curves & 3 joins to join these 4 curves.



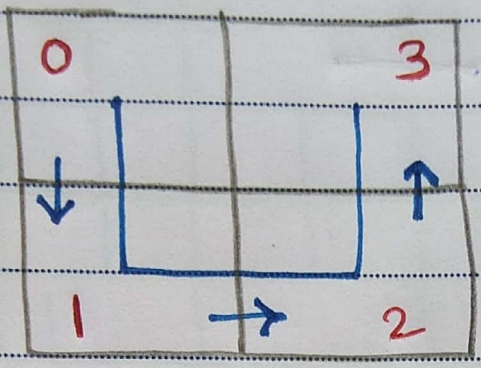
Third order Hilbert curve generated with 4 second order Hilbert curves & 3 joins.

Here we had started with first order hilbert curve with cup whose top is open i.e. ⊔

Similarly we can start with ⊓ or ⊏ or ⊐ as a first order hilbert curve.

Hilbert Curve Generation

- Consider same hilbert curve starting with cup ⊔



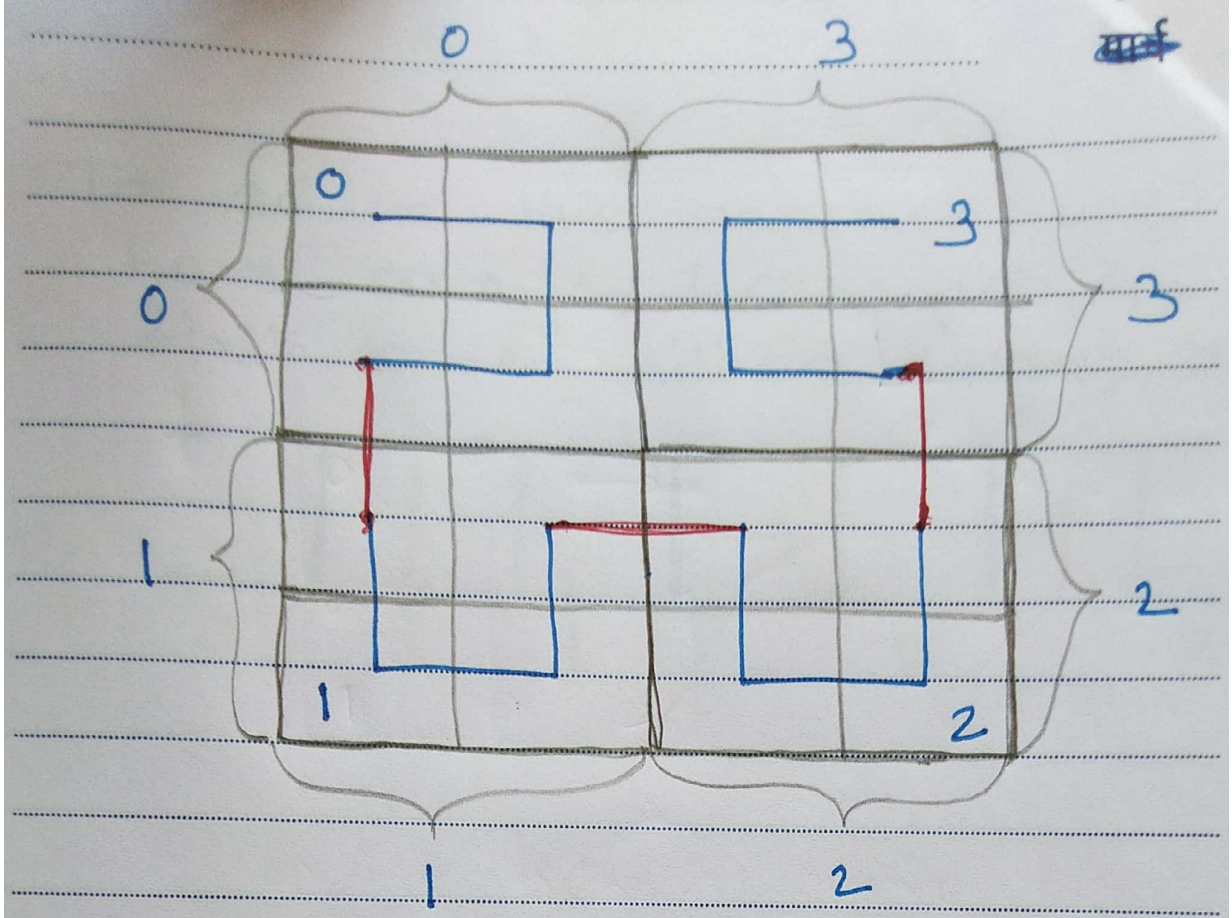
0, 1, 2, 3 are numbers given to each square of the grid.

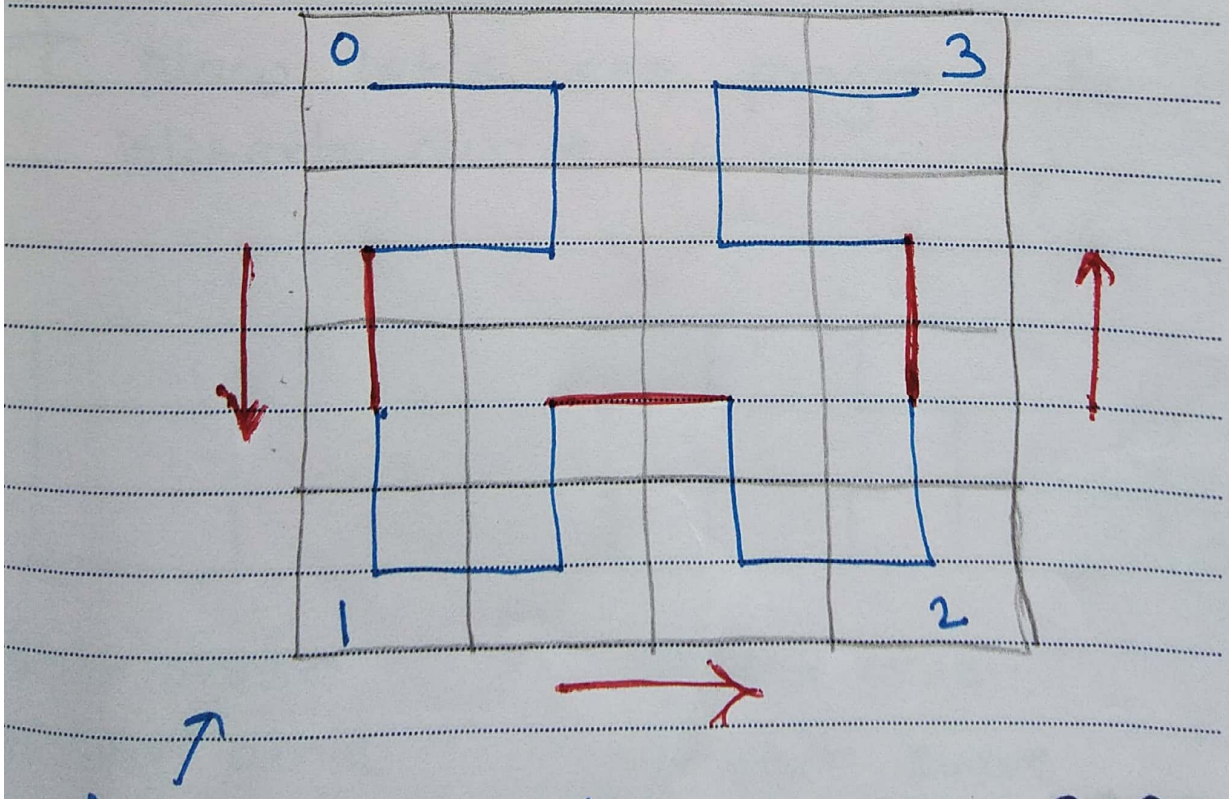
First order Hilbert curve is generated by lines visiting $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$ squares of the grid (sequence is imp)

- Second order Hilbert curve :→

Here 4 first order Hilbert curves are a arranged in 4x4 grid in such a way that it follows same sequence.
(numbers are given to each 2x2 grid in this 4x4 grid).

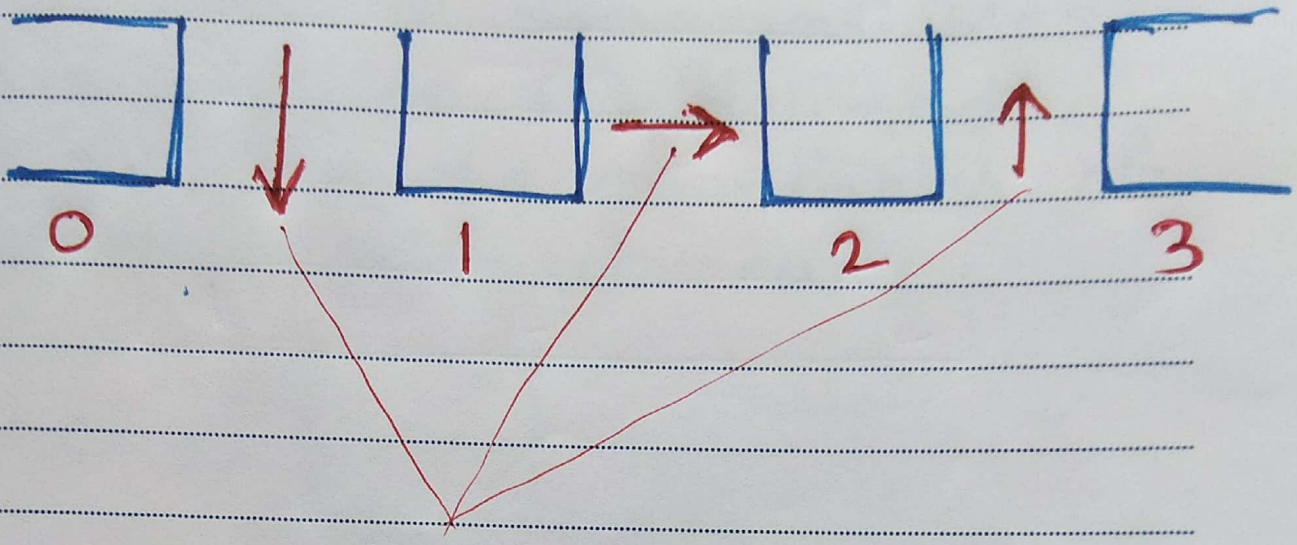
2086





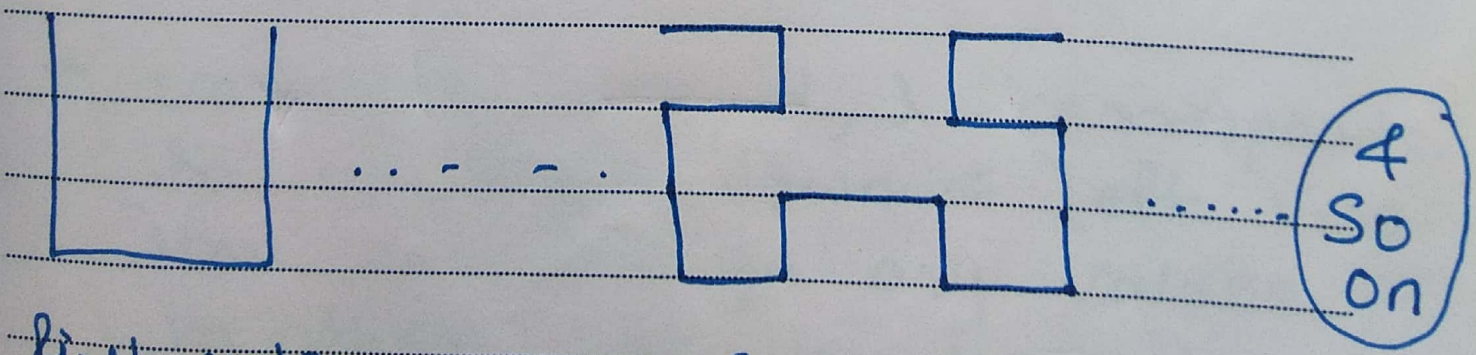
If you observe this curve, sequence 0 → 1 → 2 → 3 is maintained.

To achieve this sequence, we have arranged 4 cups i.e. 4 first order hilbert curves as



joins (along with direction)

⇒ Now let's see program for hilbert curve :-



First order hilbert curve (n=1)

Second order hilbert curve (n=2)

~~2096~~

Hilbert Curve Program

Line No. ↓

```
1   #include<iostream>
2   #include<stdio.h>
3   #include<graphics.h>
4   #include<maths.h>
5   #include<stdlib.h>
6   using namespace std;

7   void move(int j, int h, int &x, int &y)
8   {
9       if(j==1)
10          y=y-h;
11      else if(j==2)
12          x=x+h;
13      else if(j==3)
14          y=y+h;
15      else if(j==4)
16          x=x-h;
17  lineto(x,y);
18  }
```

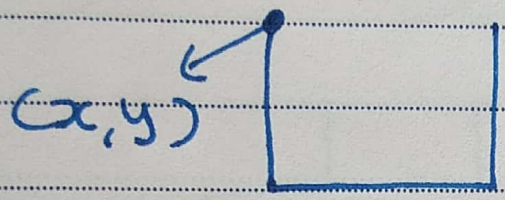
Line No

```
19  void hilbert(int d, int r, int u, int l,
                 int i, int h, int &x, int &y)
20  {
21    if (i>0)
22    {
23      i--;
24    hilbert(r,d,l,u,i,h,x,y);
25    move(d,h,x,y);
26    hilbert(d,r,u,l,i,h,x,y);
27    move(r,h,x,y);
28    hilbert(d,r,u,l,i,h,x,y);
29    move(u,h,x,y);
30    hilbert(l,u,r,d,i,h,x,y);
31    } //if
32  } //hilbert()

33  int main()
34  {
35    int x=50, y=150, h=50, u=1,
          r=2, d=3, l=4;
36  cout<<"enter the value of n";
37  cin>>n;
38  int gd=DETECT, gm;
39  initgraph(&gd, &gm, NULL);
40  moveto(x,y);
41  hilbert(d,r,u,l,n,h,x,y);
42  getch();
43  closegraph();
44    return 0;
45  }
```

⇒ We will start with **main()** function (line no:- 33 - 45)

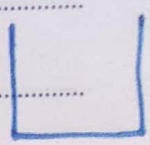
- Line no:- 35 → some variables are declared & initialized.
  - x & y are the co-ordinates of starting point of curve.



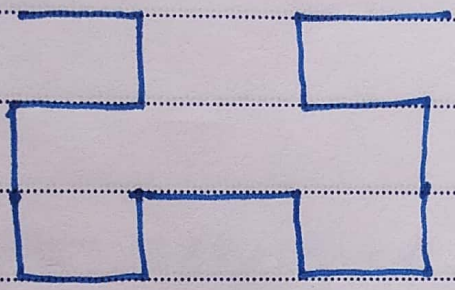
- h is the length of each line in the curve in terms of pixels. You can assign any value to h depending on what length of lines you are expecting in curve.

- variables u, r, d, l corresponds to up, right, down & left resp. You can assign any values to them but you have to use those values accordingly in 'if-else' condition in the move() function [line no:- 7 to 18] (we will discuss this later)

Line no: 36 & 37 :→ We take the value of n (i.e. the order of curve user wants) from the user.

If n=1, user is expecting → 

If n=2, user is expecting



& so on

⋮

Line no: 40 → we have studied this moveto() function. It will move the current position [which is initially (0,0)] to the parameters passed to this funct Hence current position will be moved from (0,0) to (50, 150) in this case.

x, y

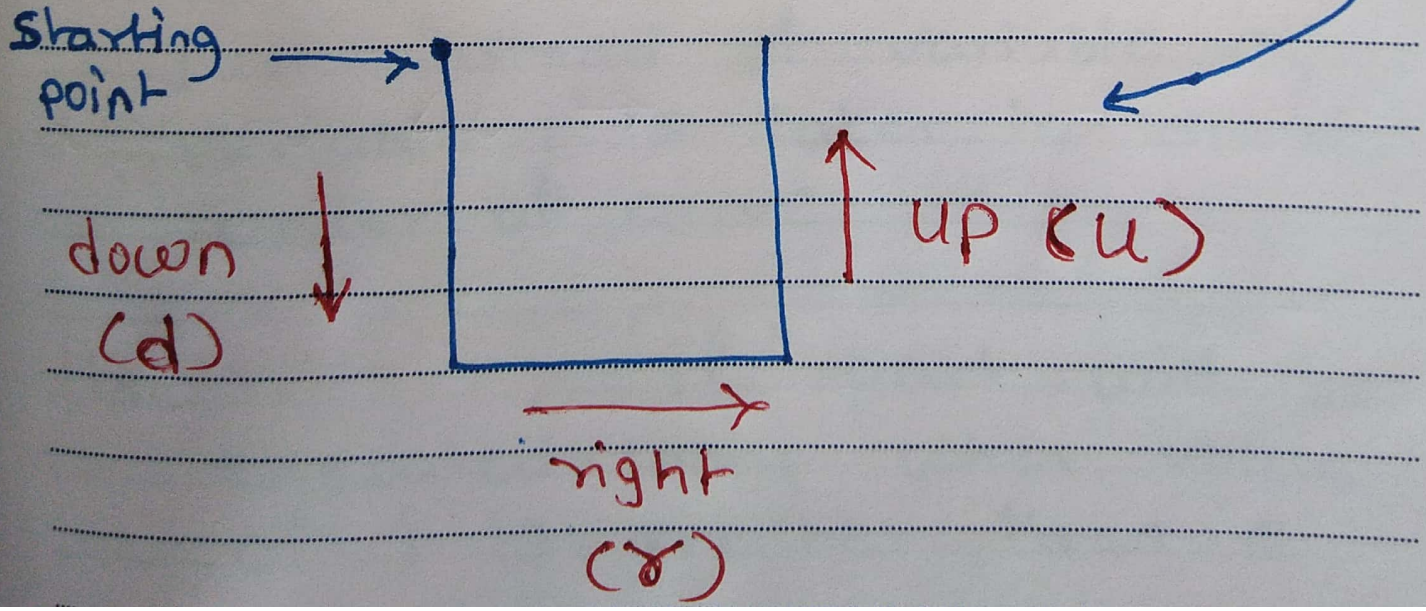
~~296~~

Line no 41 → Here we are calling hilbert() function by passing some parameters

```
hilbert(d, r, u, l, n, h, x, y);
```

down, right, up, left, order of curve, length of each line, starting point (current position)

Here ~~order~~ sequence of first 4 parameters is very important.
See we want curve like this,



So sequence of first 3 parameters is → d, x, u

& 4th parameter is remaining direction i.e. λ.

Now,

Hilbert() Function Line :- 19-32

Line no 19 → Here same sequence of ~~xxxx~~ parameters is maintained as the statement in main() function (Line no. 41).
Just instead of variable n, variable i is taken to store order of curve.

Line no 21 :→ 'if' part will be executed only when value of i is greater than 0.

So if user has given value of n (i.e. i here) as 0, then nothing will be printed on screen. This is because we will not execute 'if' part since i is not greater than 0.

So calling hilbert() function from line no. 41, will not print anything. [Since 'if' part ~~if~~ in hilbert function is entire hilbert function.].

- Now consider user has entered n = 1. (i.e. i = 1 now)

So 'if' part will execute now.

- Line no: 23 → variable 'i' will be decremented by 1.

So now i = 0

So now there are 4 hilbert() function calls (self calling func^n) in this 'if' part at line no. 24, 26, 28 & 30.

For all these 4 func^n calls, i is one parameter which has value zero now.
So now hilbert() function which is called from these line no. 24, 26, 28 & 30 will not print anything because 'if' parts in that hilbert() func^ns will not be executed since value of i has become zero now.

⇒ Hence whatever output we will get is due to ~~line no.~~ function call at line no. 25, 27 & 29. As user has entered value of n (order of curve) as 1, output ~~2016~~ expected is → ⌊⌋

This output we will get due to function calls made at line no. 25, 27 & 29. This is function call to move() funct^n.

move() funct^n line no:- 7 to 18

consider call made at line no:: 25

move(d, h, x, y)
T
down

Line no:- 7 → value of variable 'd' is ~~received~~ in variable 'j'

- d for 'down'
- so we want to ~~move~~ draw line of size 'h' in downward direction
- remember value stored in variable 'd' is 3.
- Therefore at line no. 13 we have written condition if j==3 then y = y+h

y coordinate is increased by h means we are moving downwards by h pixels

So now value of x is as it is i.e. 50. But value of y is increased by 'h=50' i.e. 150+50=200 (y+h=y)

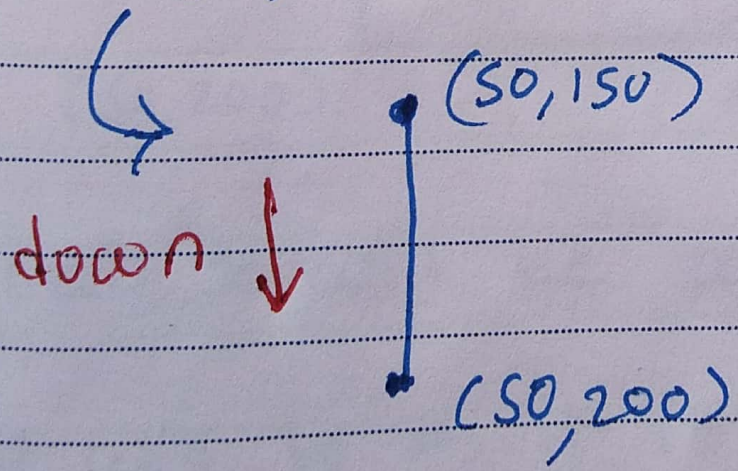
∴ x=50, y=200 now.

Now line no. 17 will be executed

lineto(x,y);

50 → 200

∴ Line is drawn from current position (50,150) to (x,y) i.e. (50,200)



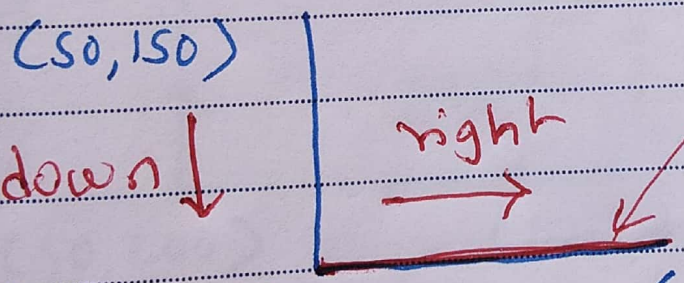
Now see function call at line no: 27

move (r, h, x, y).

r → right

So at line no. 7, value of r (right) i.e. 2 is received in variable j.

Now condition line no. 11 will be satisfied. So x = x+h is done as we wanted to draw line from left to right.



∴ new x will be x+h i.e. 50+50 = 100
x h

y will be as it is.

∴ now due to line no. 17, line is drawn from current position (50,200) to (100,200).

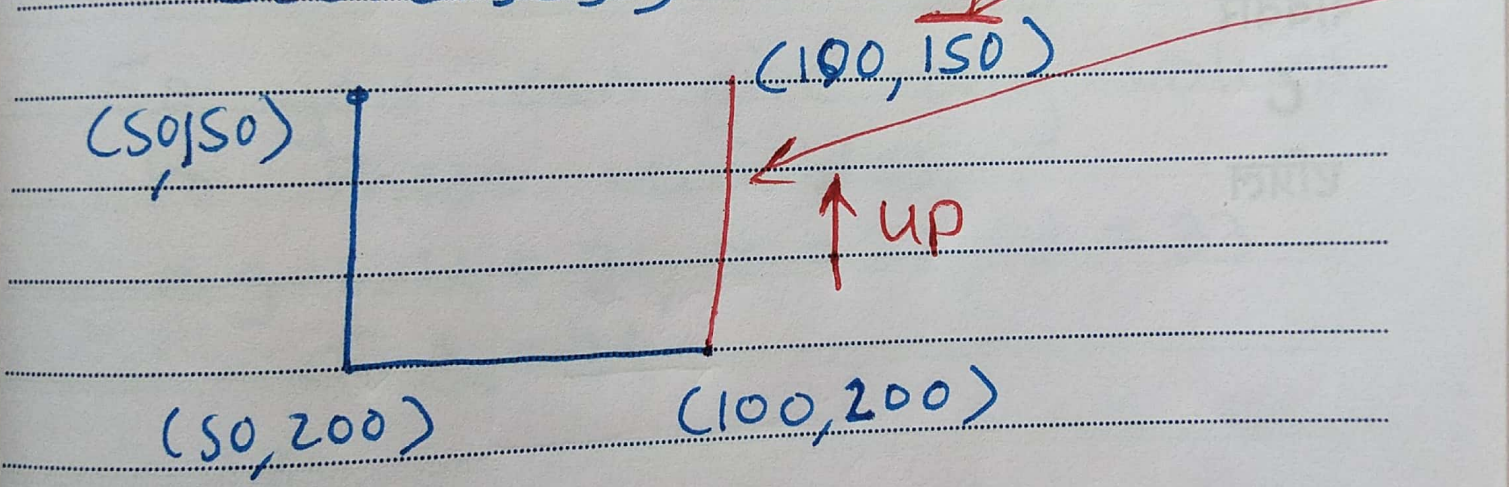
Now, due to function call at line-29

move(u, h, x, y);

up → [u=1. so cond^n at line no. 9 is satisfied]

line is drawn in upward direction due to line no 9 & 10

- y will be decreased by 50 pixels so we move upward (as y decreases).



⇒ This is the significance of move() function. Actually hilbert curve is printed because of move() funct^n.

In this move() function, all 'if' conditions are such that you can move in up, down, left & right direction.

↳ These cond^ are set depending on the values given to variables u, r, d & l in main() function.

So you can give any values to these variables

e.g. u = 31, r = 32, d = 33 & l = 34

↙

For this you have to change cond^s in line no:- 9, 11, 13, 15 as

if (j == 31) ← line 9
if (j == 32) ← line 11
if (j == 33) ← line 13
if (j == 34) ← line 15

This is how first order Hilbert curve is generated.

- Now suppose user has entered value of $n = 2$.

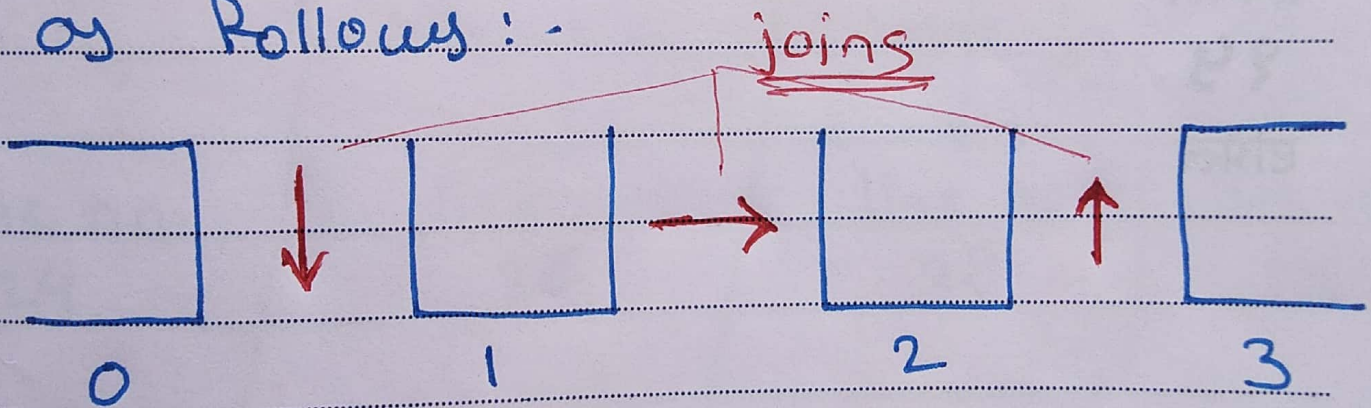
Due to line no. 23, n will be decremented by 1 $\therefore n = 2 - 1 = 1$

- But now funct^n calls at line no. 24, 26, 28 & 30 will be having role in printing the lines.
- These are the self calling functions that will work similarly what we have seen now.

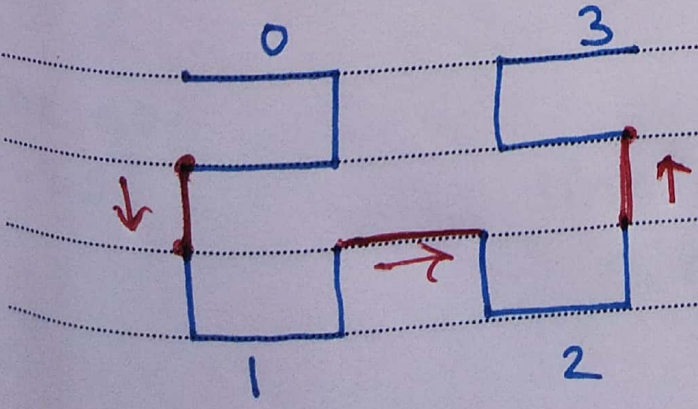
- If you look carefully, first 4 parameters that are passed at line no. 24, 26, 28 & 30 has different sequence.

⇒ Sequence of these parameters is decided as follows :-

we have seen, if we want to generate hilbert curve whose first order hilbert cure is ⊔. To get second order hilbert curve, we have to arrage 4 first order hilbert curves as follows :-

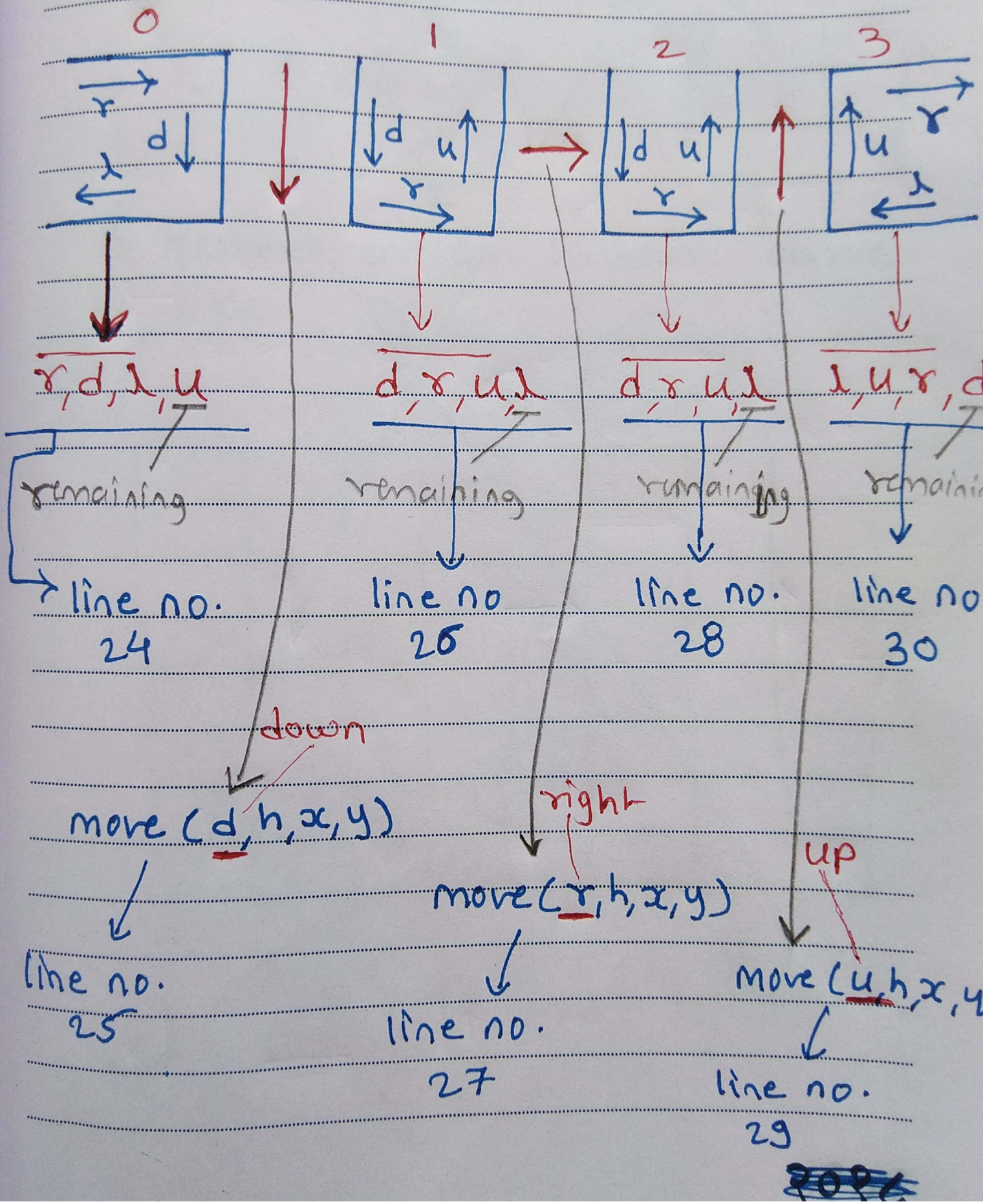


so that we will get.

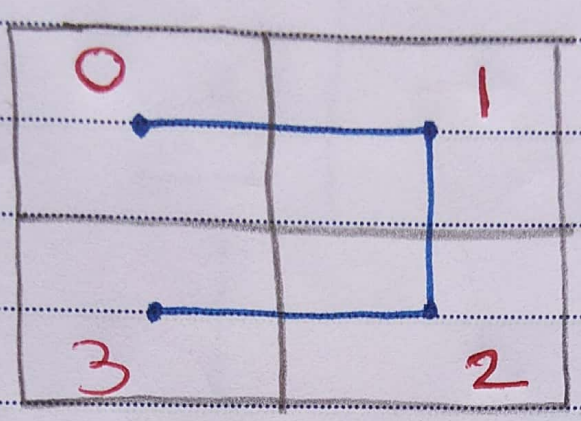


2026

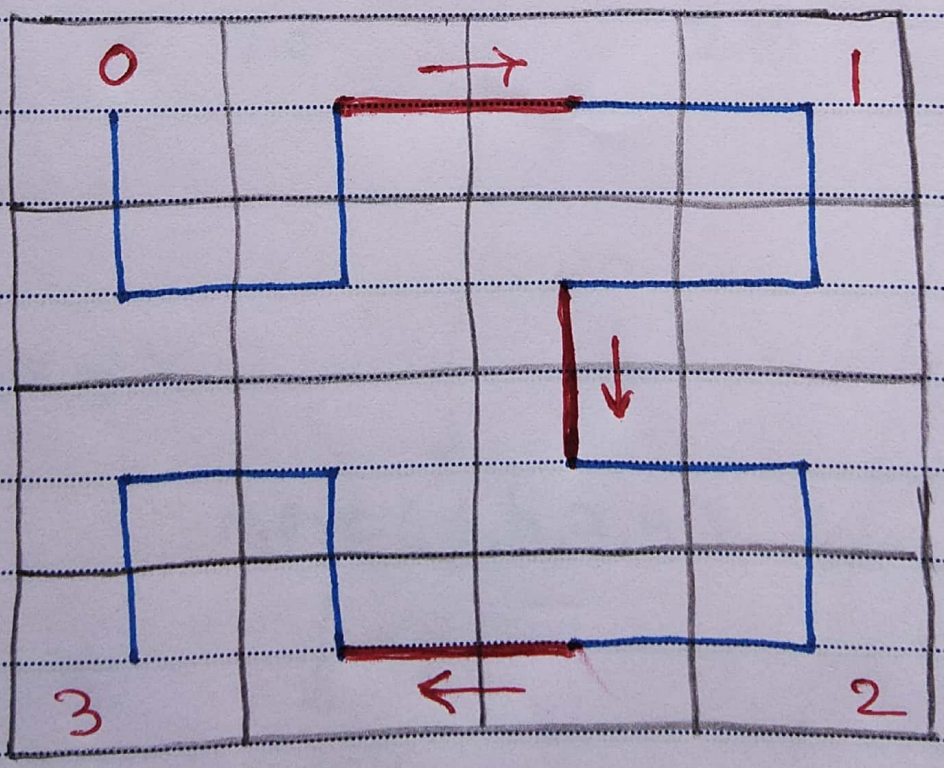
Again I am writing the sequence :-



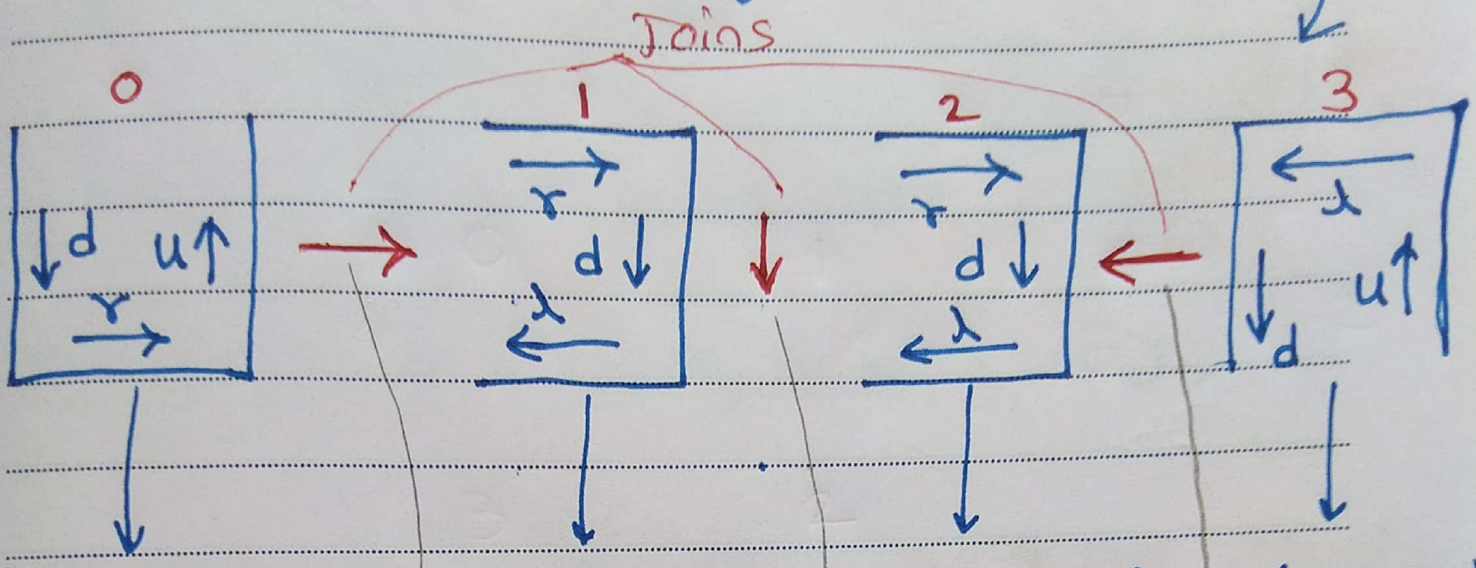
* If we want first order Hilbert curve like ↓



& second order Hilbert curve like ↓



4 first ordered hilbert curves will be arranged like this



(d,r,u,l) (r,d,l,u) (r,d,l,u) (u,l,d,r)

line no. 24 | line no. 26 | line no. 28 | line no. 30

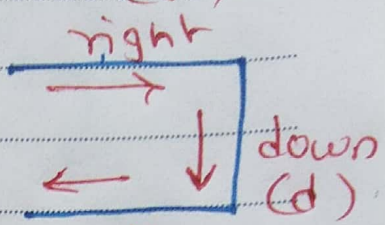
right | down | left

move(r,h,x,y) | move(d,h,x,y) | move(l,h,x,y)

line no. 25 | line no. 27 | line no. 29

So in the program for this type of Hilbert curve only some changes will be there as follows:

बुधवार
24
एप्रिल



⇒ main() Funct^

Funct^ call will be like

```
hilbert (r,d,l,u,n,h,x,y);
```

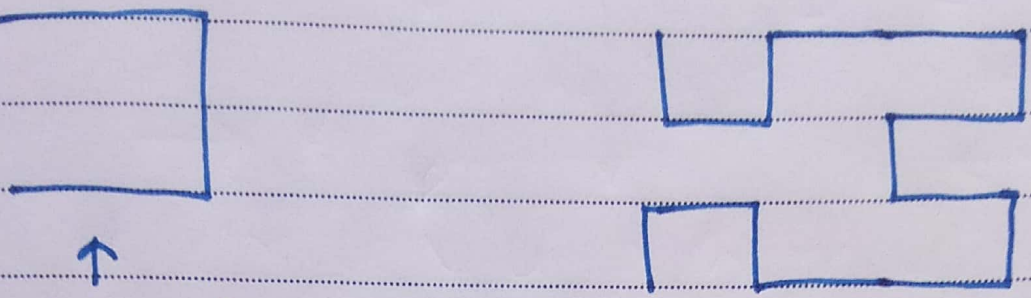
⇒ Hilbert() Funct^

sequence should same

```
void hilbert (int r, int d, int l, int u,
              int i, int h, int *x, int *y)
{
  if (i>0)
{ i--;
  hilbert (d,r,u,l,i,h,x,y);
  move (r,h,x,y);
  hilbert (r,d,l,u,i,h,x,y);
  move (d,h,x,y);
  hilbert (r,d,l,u,i,h,x,y);
  move (l,h,x,y);
  hilbert (u,l,d,r,i,h,x,y);
}
}
```

2026

Do only above changes you will get hilbert curve like ↓

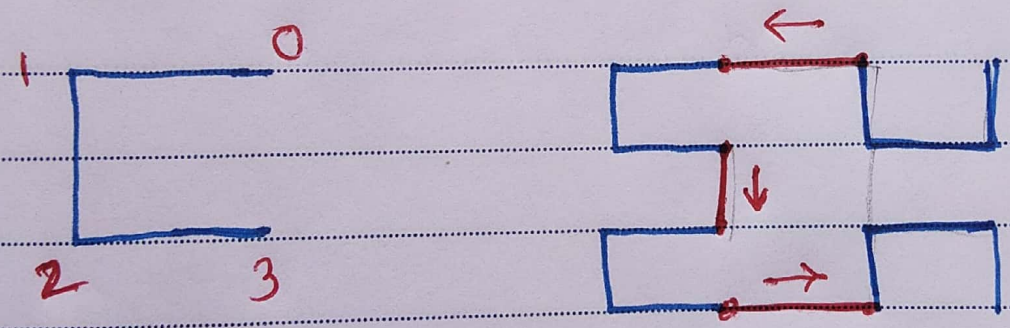


First order

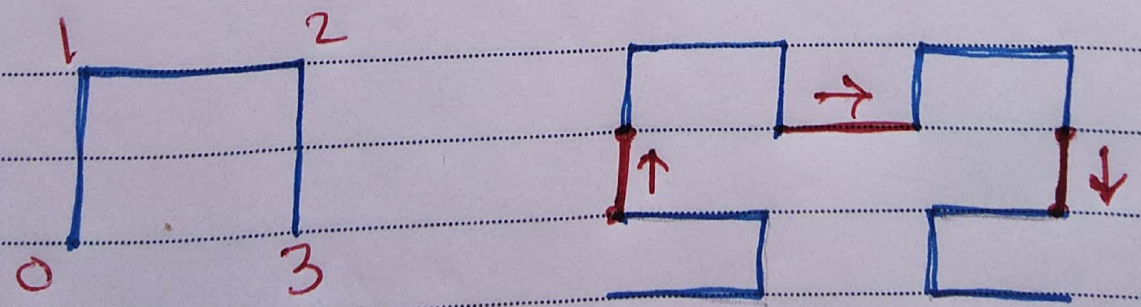
second order.

Similarly you can generate following types of hilbert curves :-

(3)



(4)



Try above 2 types of curves as a home work.

2022