```python
"""Hilbert Curve"""
# N.B. This is a bit of a crude version as I couldn't find my old implementation
from calendar import leapdays
import turtle
count = 0
wn = turtle.Screen()
turtle.screensize(700,700)
turtle.hideturtle()
turtle.penup()
turtle.backward(300)
turtle.left(90)
turtle.backward(300)
turtle.pendown()

def segments(count):
    if(count > 1):
        return 2*segments(count-1) +1
    else:
        return 3

def seedL(length):
    turtle.forward(length)
    turtle.left(90)
    turtle.forward(length)
    turtle.left(90)
    turtle.forward(length)

def seedR(length):
    turtle.forward(length)
    turtle.right(90)
    turtle.forward(length)
    turtle.right(90)
    turtle.forward(length)

def recursiveR(count, length):
    count -= 1
    rec = length/segments(count)
    rec_ = segments(count-1)*rec
    len = length/3
    if count>1:
        turtle.right(90)
        recursiveL(count,rec_)
        turtle.right(90)
        turtle.forward(rec)
        recursiveR(count,rec_)
        turtle.left(90)
        turtle.forward(rec)
        turtle.left(90)
        recursiveR(count,rec_)
        turtle.forward(rec)
        turtle.right(90)
        recursiveL(count,rec_)
        turtle.right(90)
    else:
        turtle.right(90)
        seedL(len)
        turtle.right(90)
        turtle.forward(len)
        seedR(len)
        turtle.left(90)
        turtle.forward(len)
        turtle.left(90)
        seedR(len)
        turtle.forward(len)
        turtle.right(90)
        seedL(len)
        turtle.right(90)

def recursiveL(count, length):
    count -= 1
    rec = length/segments(count)
    rec_ = segments(count-1)*rec
    len = length/3
    if count>1:
        turtle.left(90)
        recursiveR(count,rec_)
        turtle.left(90)
        turtle.forward(rec)
        recursiveL(count,rec_)
        turtle.right(90)
        turtle.forward(rec)
        turtle.right(90)
        recursiveL(count,rec_)
        turtle.forward(rec)
        turtle.left(90)
        recursiveR(count,rec_)
        turtle.left(90)
    else:
        turtle.left(90)
        seedR(len)
        turtle.left(90)
        turtle.forward(len)
        seedL(len)
        turtle.right(90)
        turtle.forward(len)
        turtle.right(90)
        seedL(len)
        turtle.forward(len)
        turtle.left(90)
        seedR(len)
        turtle.left(90)

try:
    count = int(input("Psudeo Order: "))
    recursiveR(count, 600)
    print("Finished")
except ValueError:
    raise ValueError("Order must be positive interger")
```

```python
"""Koch Snoflake"""
import turtle
count = 0
wn = turtle.Screen()
turtle.hideturtle()
turtle.penup()
turtle.backward(400)
turtle.pendown()

def drawseed(length):
    len = length/3
    turtle.forward(len)
    turtle.left(60)
    turtle.forward(len)
    turtle.right(120)
    turtle.forward(len)
    turtle.left(60)
    turtle.forward(len)

def recursive(count, length):
    count -= 1
    len = length/3
    if count>1:
        recursive(count,len)
        turtle.left(60)
        recursive(count,len)
        turtle.right(120)
        recursive(count,len)
        turtle.left(60)
        recursive(count,len)
    else:
        drawseed(len)

try:
    count = int(input("Psudeo Order: "))
    recursive(count, 2400)
    print("Finished")
except ValueError:
    raise ValueError("Order must be positive interger")

```

```python
"""Dragon Curve"""
import sys, turtle
Gcount = 0
wn = turtle.Screen()
turtle.hideturtle()
turtle.penup()
turtle.backward(250)
turtle.left(90)
turtle.forward(200)
turtle.right(135)
turtle.pendown()

def rRecursive(count, length):
    count -= 1
    newLength = ((length**2)/2)**0.5
    if count > 0:
        turtle.left(45)
        lRecursive(count, newLength)
        turtle.right(90)
        rRecursive(count, newLength)
        turtle.left(45)
    else:
        turtle.left(45)
        turtle.forward(length)
        turtle.right(90)
        turtle.forward(length)
        turtle.left(45)

def lRecursive(count, length):
    count -= 1
    newLength = ((length**2)/2)**0.5
    if count > 0:
        turtle.right(45)
        lRecursive(count, newLength)
        turtle.left(90)
        rRecursive(count, newLength)
        turtle.right(45)
    else:
        turtle.right(45)
        turtle.forward(length)
        turtle.left(90)
        turtle.forward(length)
        turtle.right(45)

try:
    Gcount = int(input("Psudeo Order: "))
    filename = "Order{0}PseudoDragonCurve.ps".format(Gcount)
    print(filename)
    rRecursive(Gcount, 350)
    print("Finished")
    turtle.getscreen().getcanvas().postscript(file=filename)
except ValueError:
    raise ValueError("Order must be positive interger")

sys.exit()
```

```python
"""Sierpinksi Triangle"""
import sys, turtle
Gcount = 0
wn = turtle.Screen()
turtle.hideturtle()
turtle.penup()
turtle.backward(350)
turtle.left(90)
turtle.backward(300)
turtle.right(30)
turtle.pendown()

def seed(sideLength):
    for i in range(3):
        turtle.forward(sideLength)
        turtle.right(120)

def recursive(count,sideLength):
    count -= 1
    if count > 0:
        recursive(count,sideLength/2)
        turtle.forward(sideLength/2)
        recursive(count,sideLength/2)
        turtle.right(120)
        turtle.forward(sideLength/2)
        turtle.left(120)
        recursive(count,sideLength/2)
        turtle.left(120)
        turtle.forward(sideLength/2)
        turtle.right(120)
    else:
        seed(sideLength)

try:
    Gcount = int(input("Psudeo Order: "))
    filename = "Order{0}PseudoSierpinskiTriangle.ps".format(Gcount)
    print(filename)
    recursive(Gcount, 700)
    print("Finished")
    turtle.getscreen().getcanvas().postscript(file=filename)
except ValueError:
    raise ValueError("Order must be positive interger")

sys.exit()
```