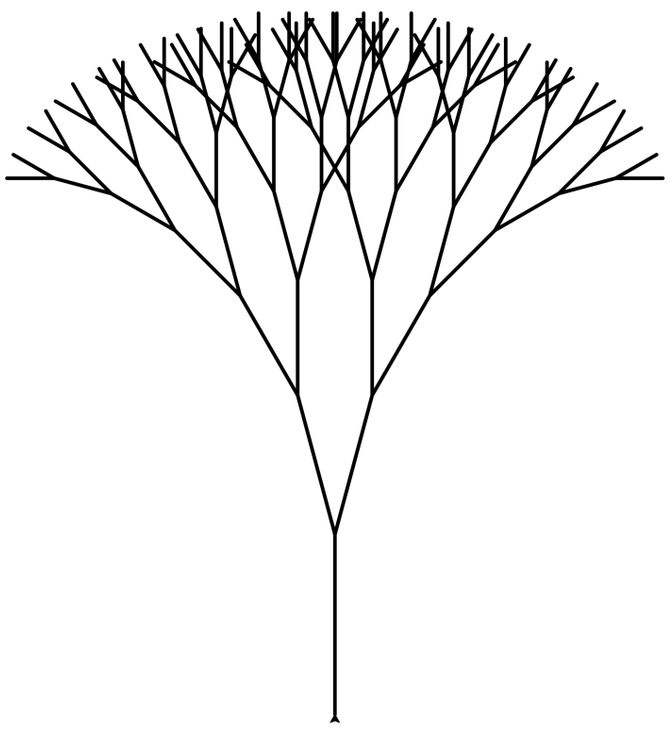
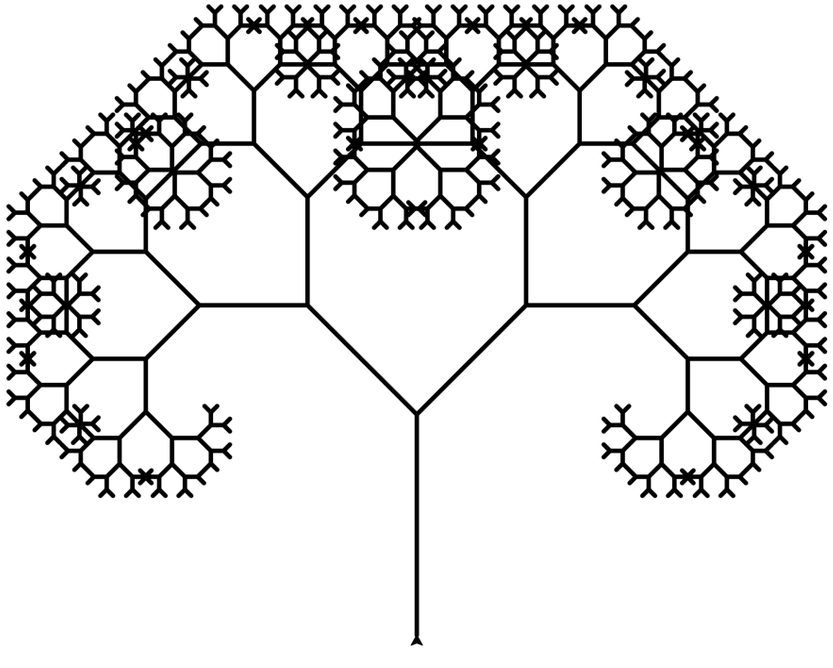
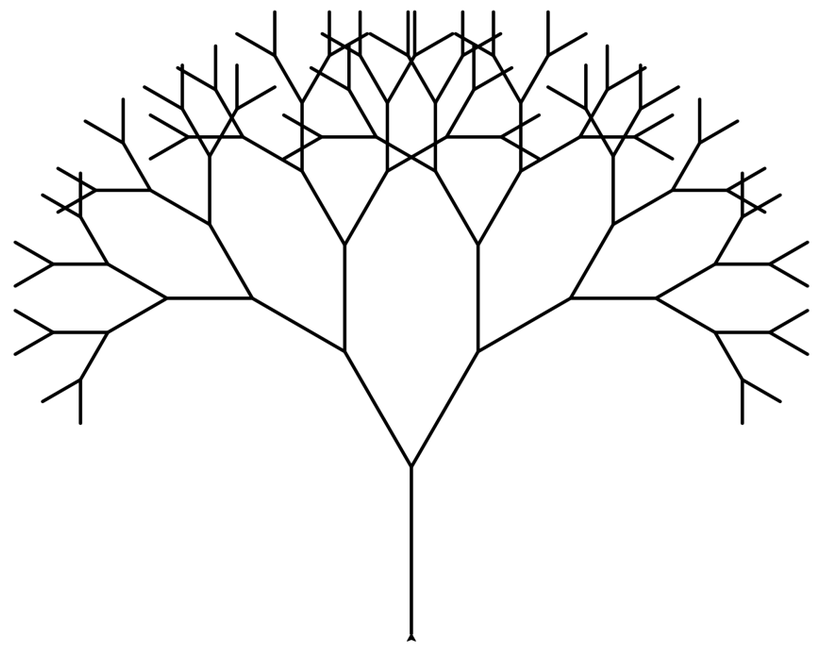
Graphical Recursion II

Trees







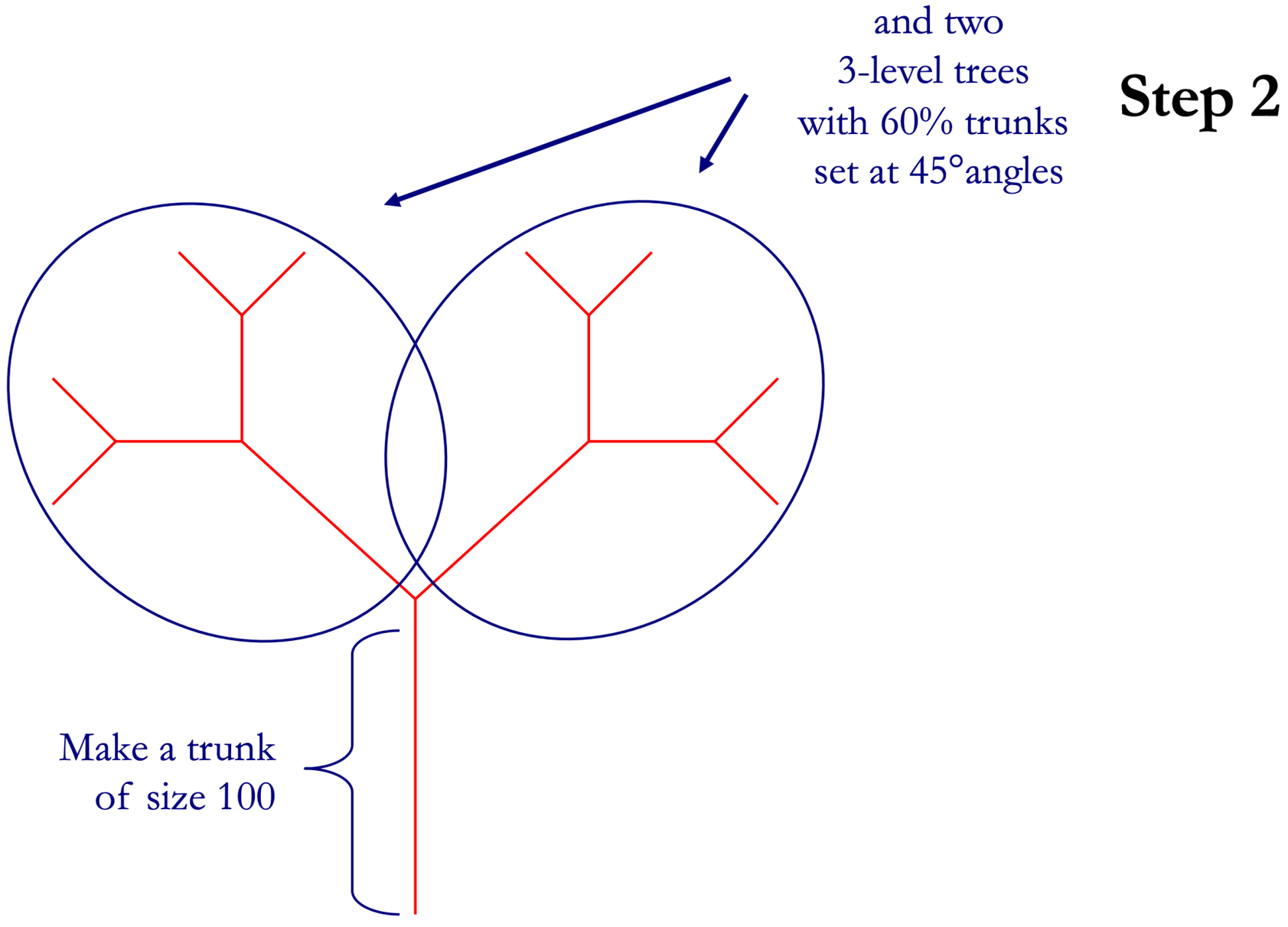
Recursive Trees

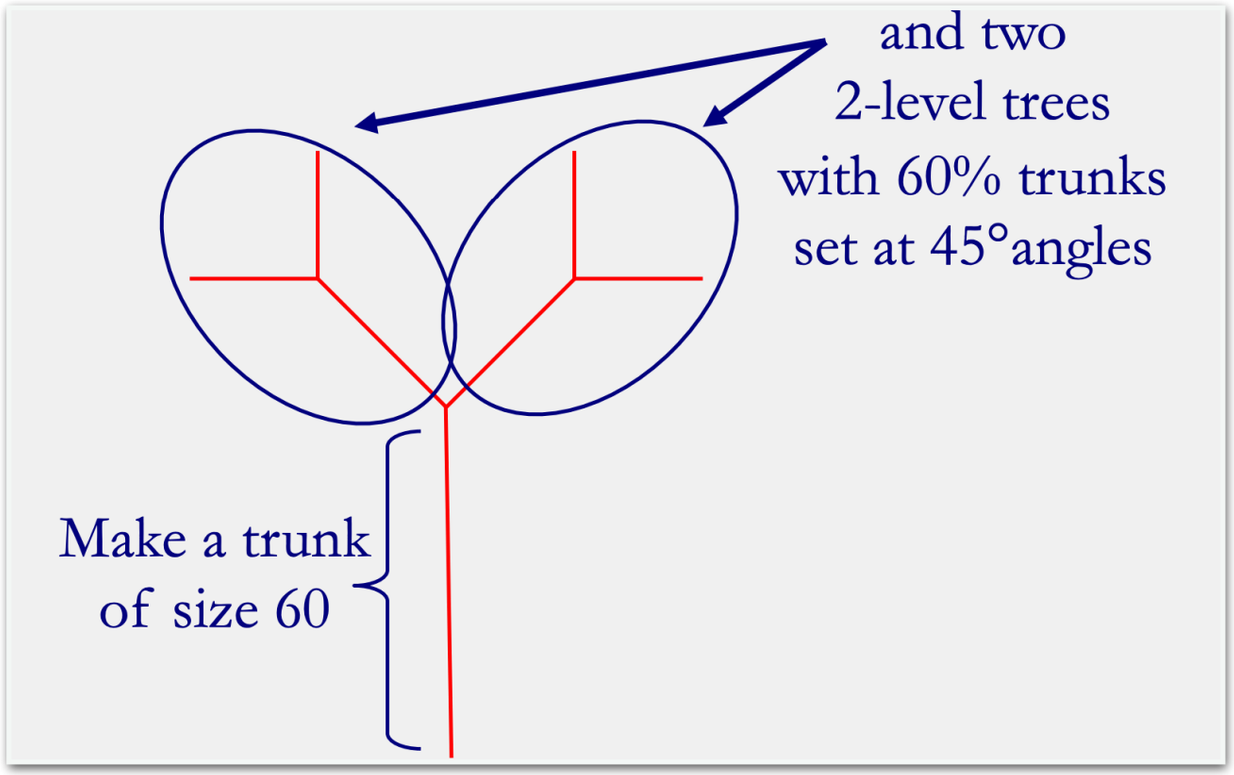
`tree(trunkLen, levels, angle, shrinkFactor)`

- `trunkLen`: is the length of the base trunk of the tree
- `levels`: is the number of branches on any path from root to leaf
- `angle`: is the angle from the trunk of the right and left branches
- `shrinkFactor`: is the factor by which the trunkLen of the branches goes down by

How to make a 4-level tree?

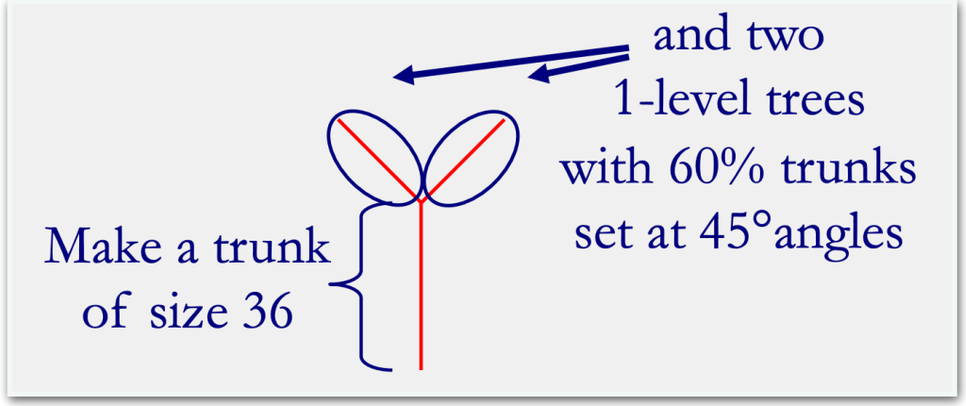
```
tree(100, 4, 45, 0.6)
```





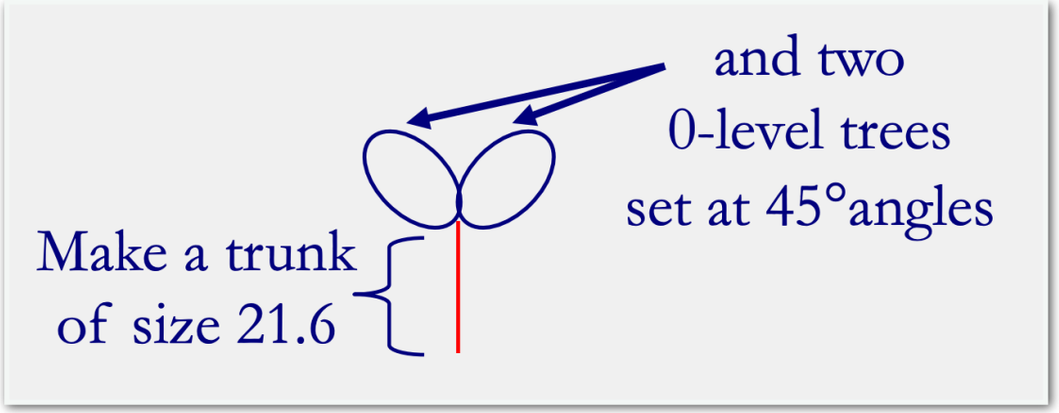

```
tree(60, 3, 45, 0.6)
```

How to make a 3-level tree?




```
tree(36, 2, 45, 0.6)
```

How to make a 2-level tree?




```
tree(21.6, 1, 45, 0.6)
```

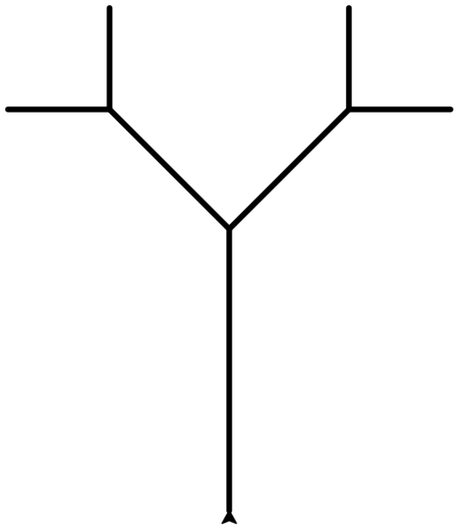
How to make a 1-level tree?

Do nothing!

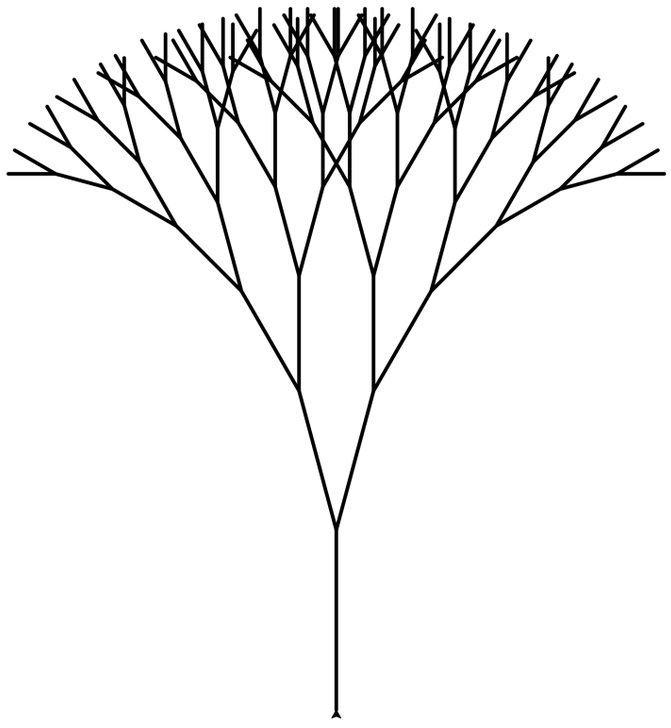
```
tree(12.96, 0, 45, 0.6)
```

How to make a 0-level tree?

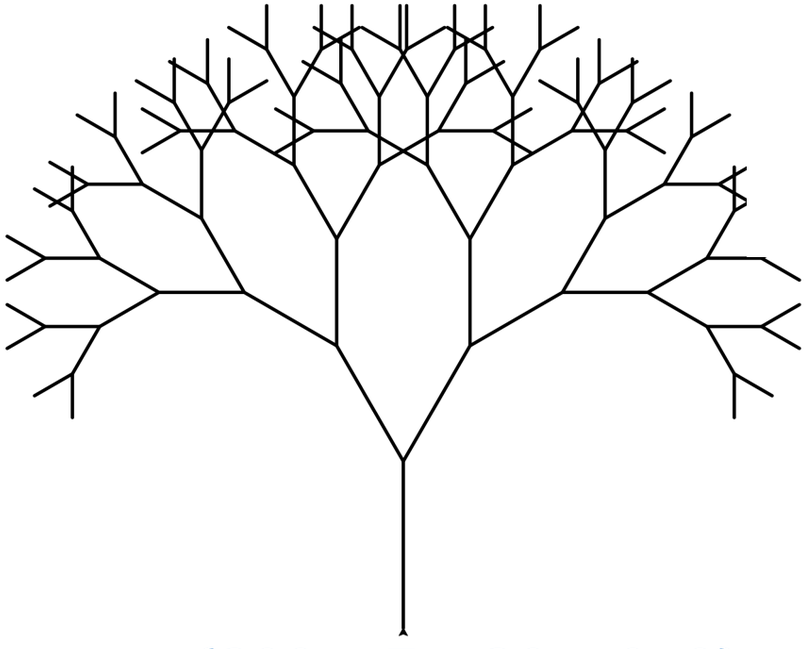
Trees



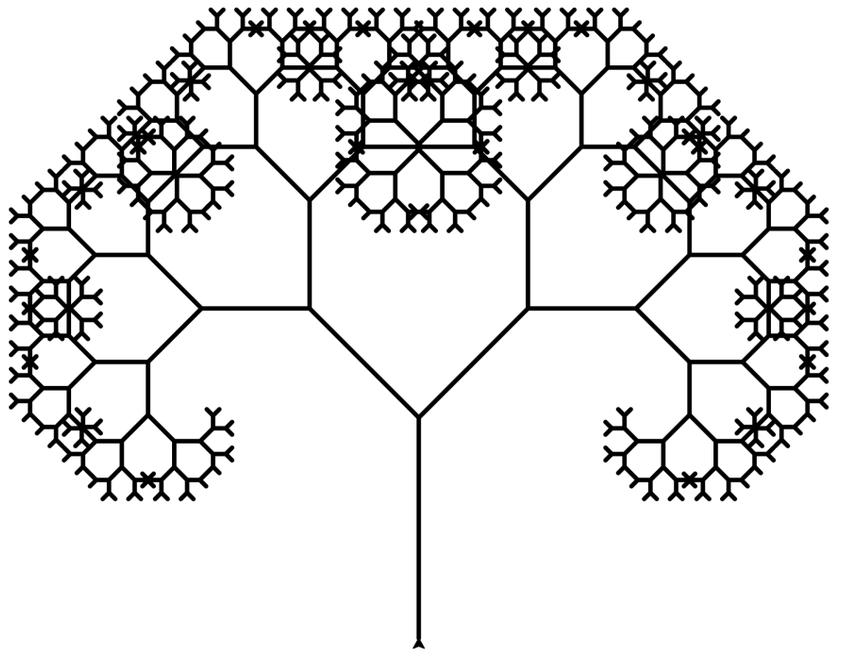
tree(200, 3, 45, 0.6)



tree(200, 7, 15, 0.8)



tree(200, 7, 30, 0.8)



tree(200, 10, 45, 0.7)

Function Frame Model to Understand `tree(60, 3, 45, 0.6)`

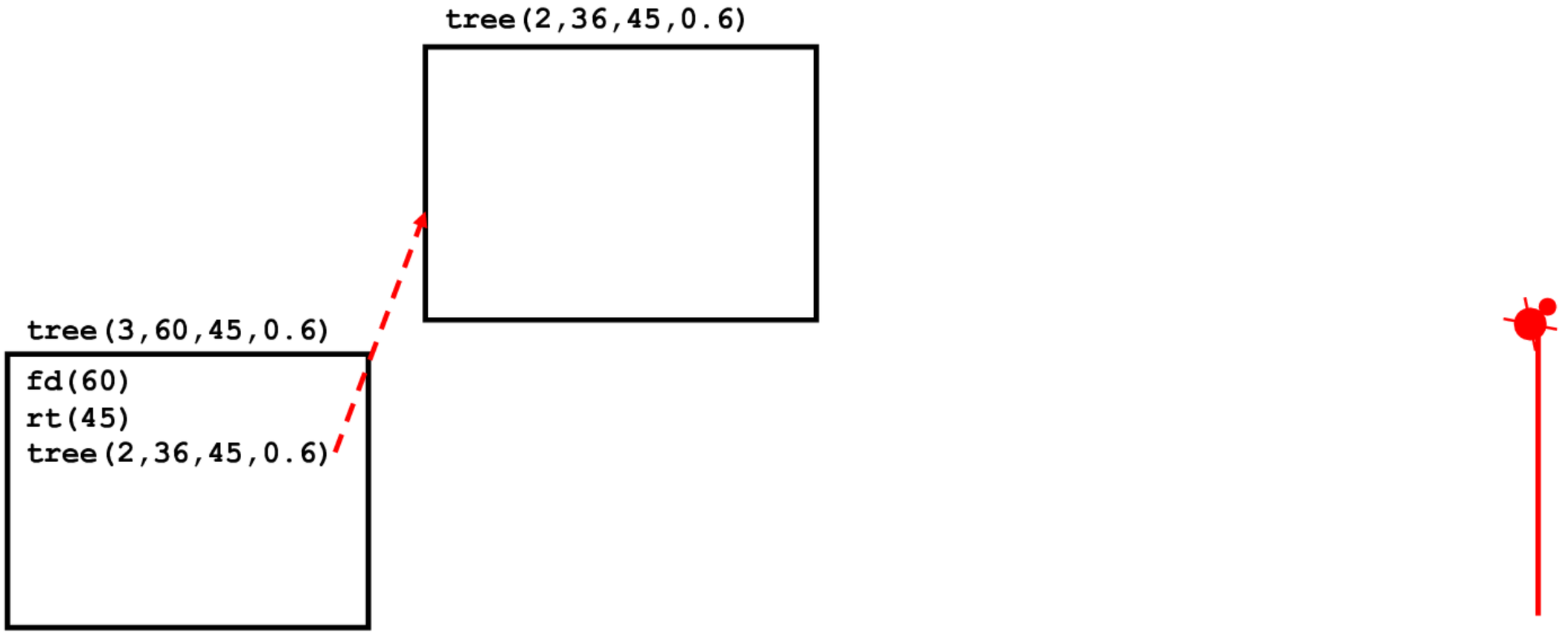
Draw trunk and turn to draw level 2 tree

```
tree(3,60,45,0.6)
```

```
fd(60)
rt(45)
```

Source: http://cs111.wellesley.edu/spring19

Begin recursive invocation to draw level 2 tree



Source: http://cs111.wellesley.edu/spring19

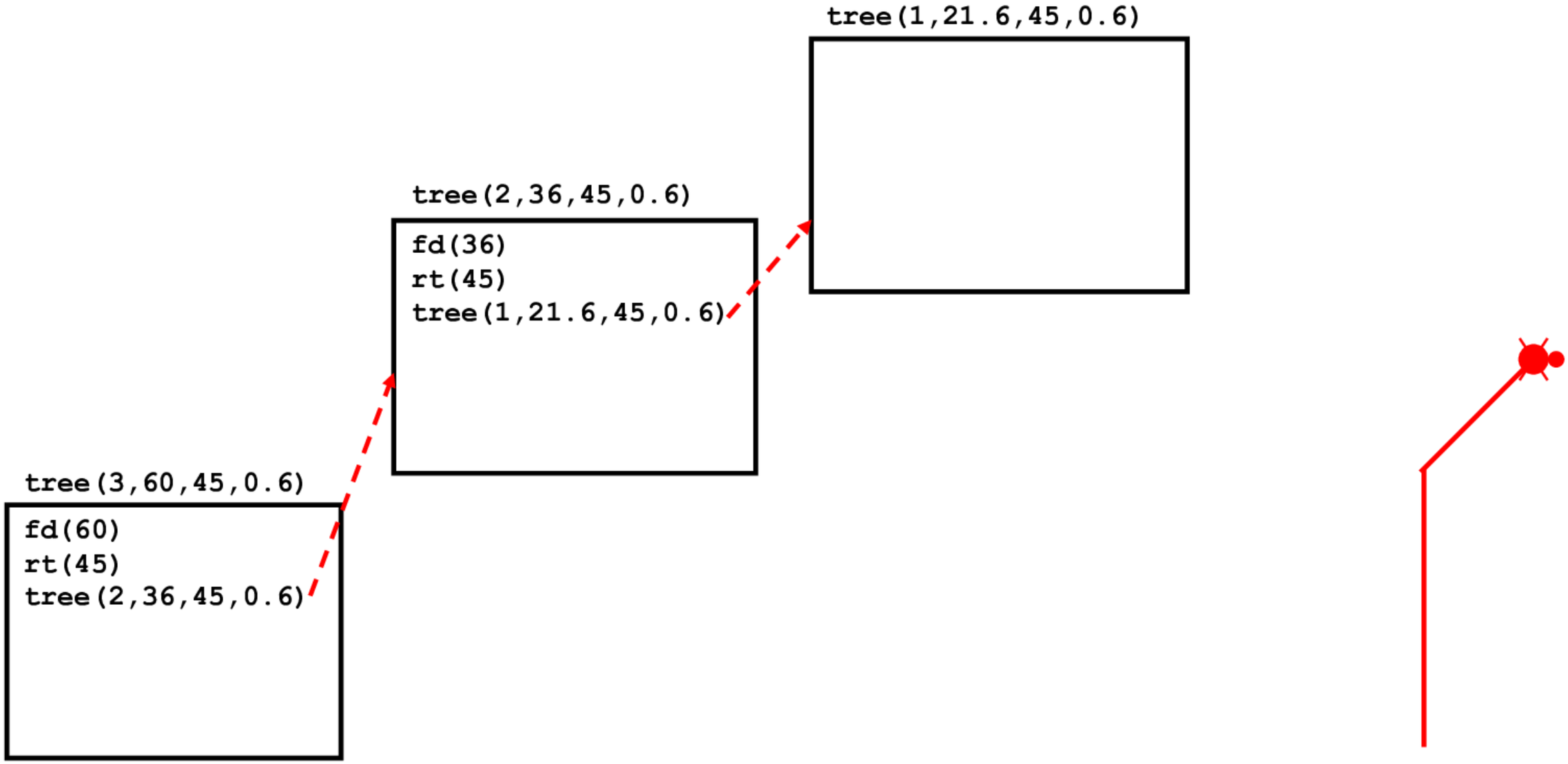
Draw trunk and turn to draw level 1 tree

```
tree(2,36,45,0.6)
fd(36)
rt(45)
```

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
```

Source: http://cs111.wellesley.edu/spring19

Begin recursive invocation to draw level 1 tree



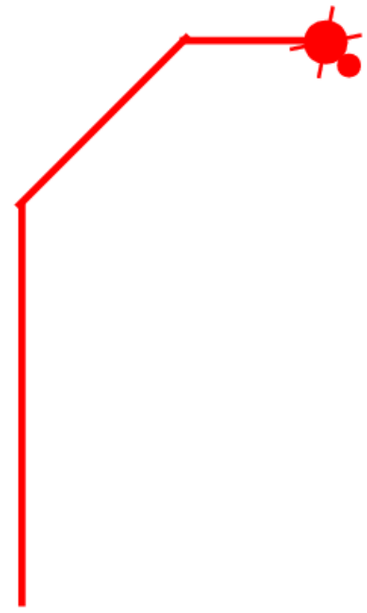
Source: http://cs111.wellesley.edu/spring19

Draw trunk and turn to draw level 0 tree

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
```

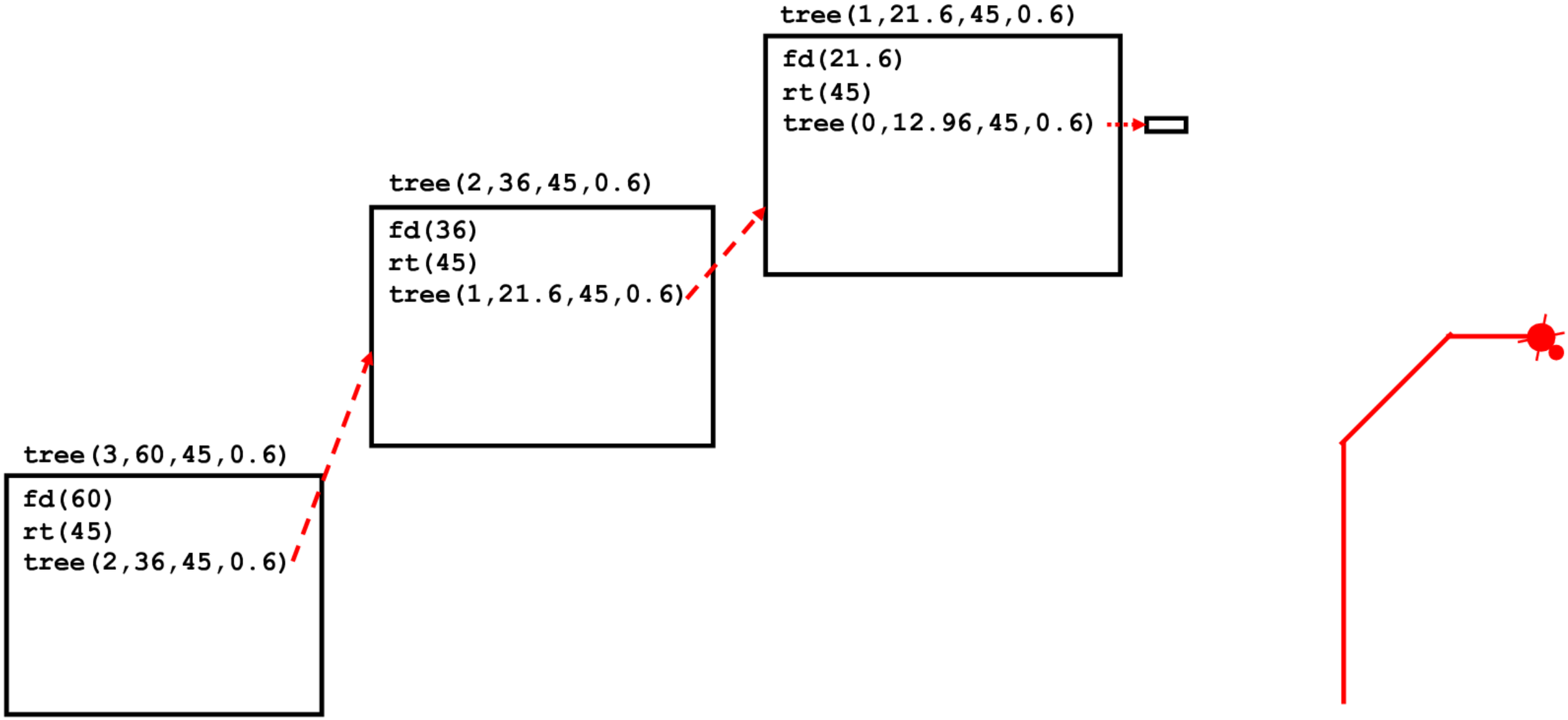
```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
```



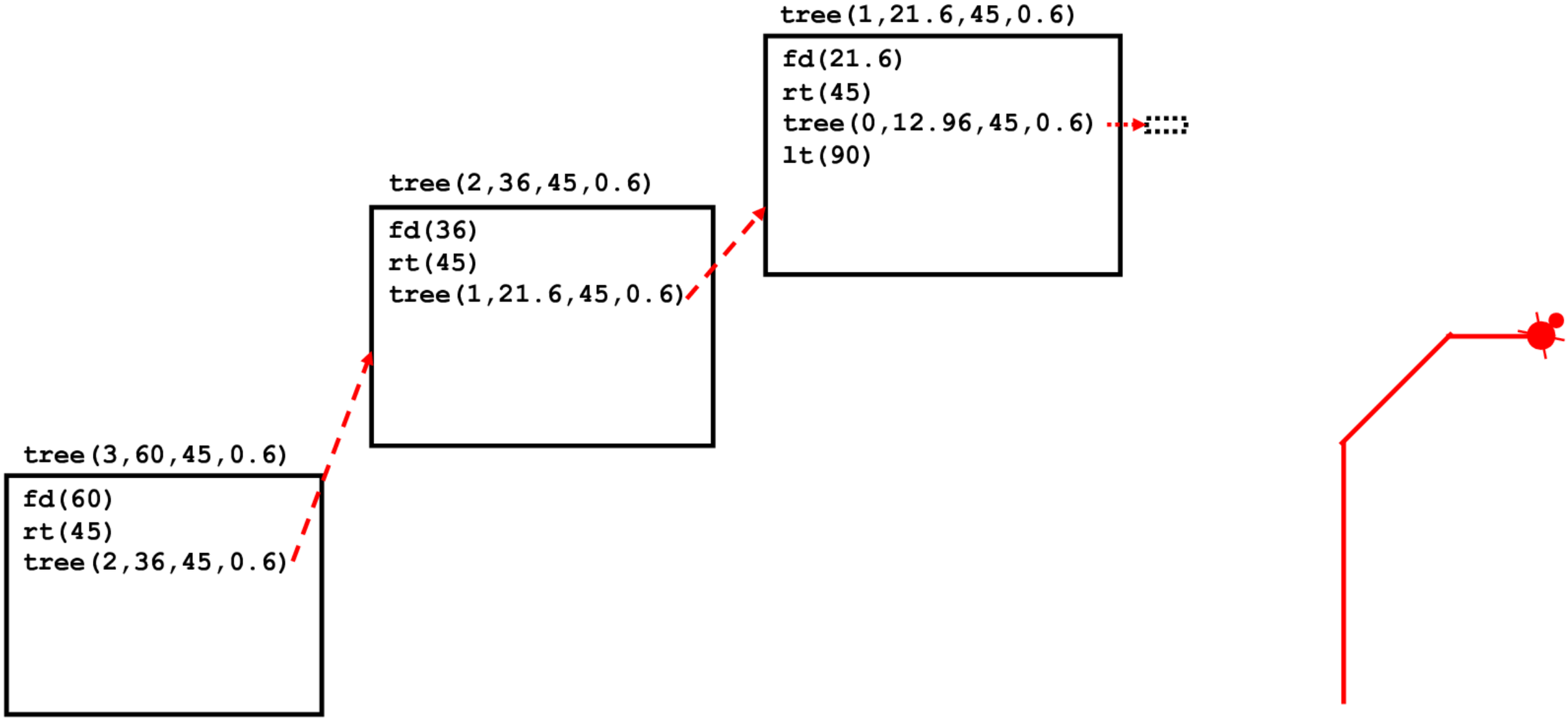
Source: http://cs111.wellesley.edu/spring19

Begin recursive invocation to draw level 0 tree



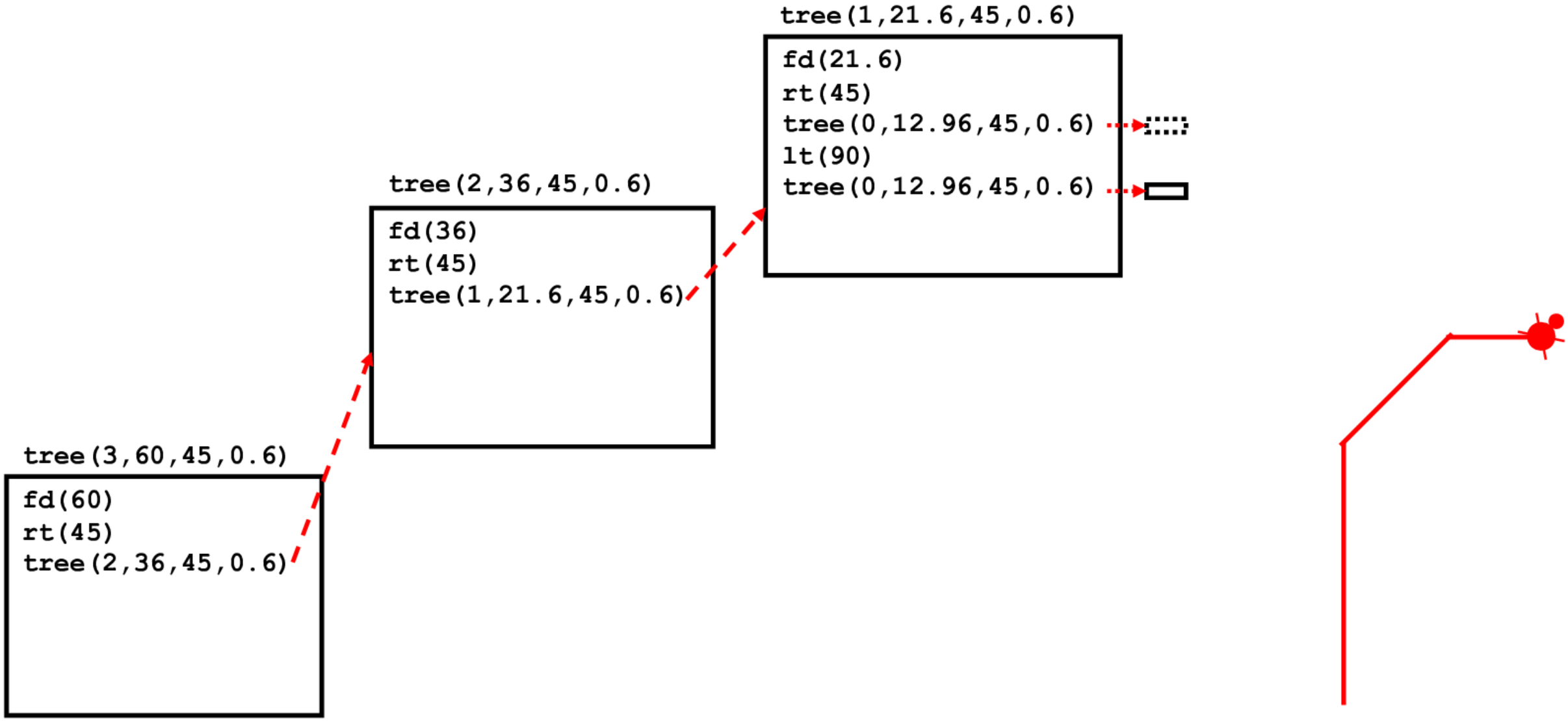
Source: http://cs111.wellesley.edu/spring19

Complete level 0 tree and turn to draw another level 0 tree



Source: http://cs111.wellesley.edu/spring19

Begin recursive invocation to draw level 0 tree



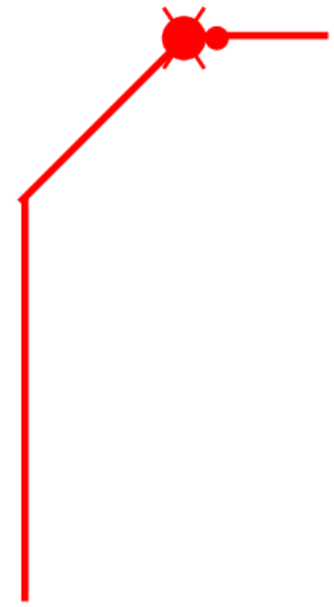
Source: http://cs111.wellesley.edu/spring19

Complete level 0 tree and return to starting position of level 1 tree

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

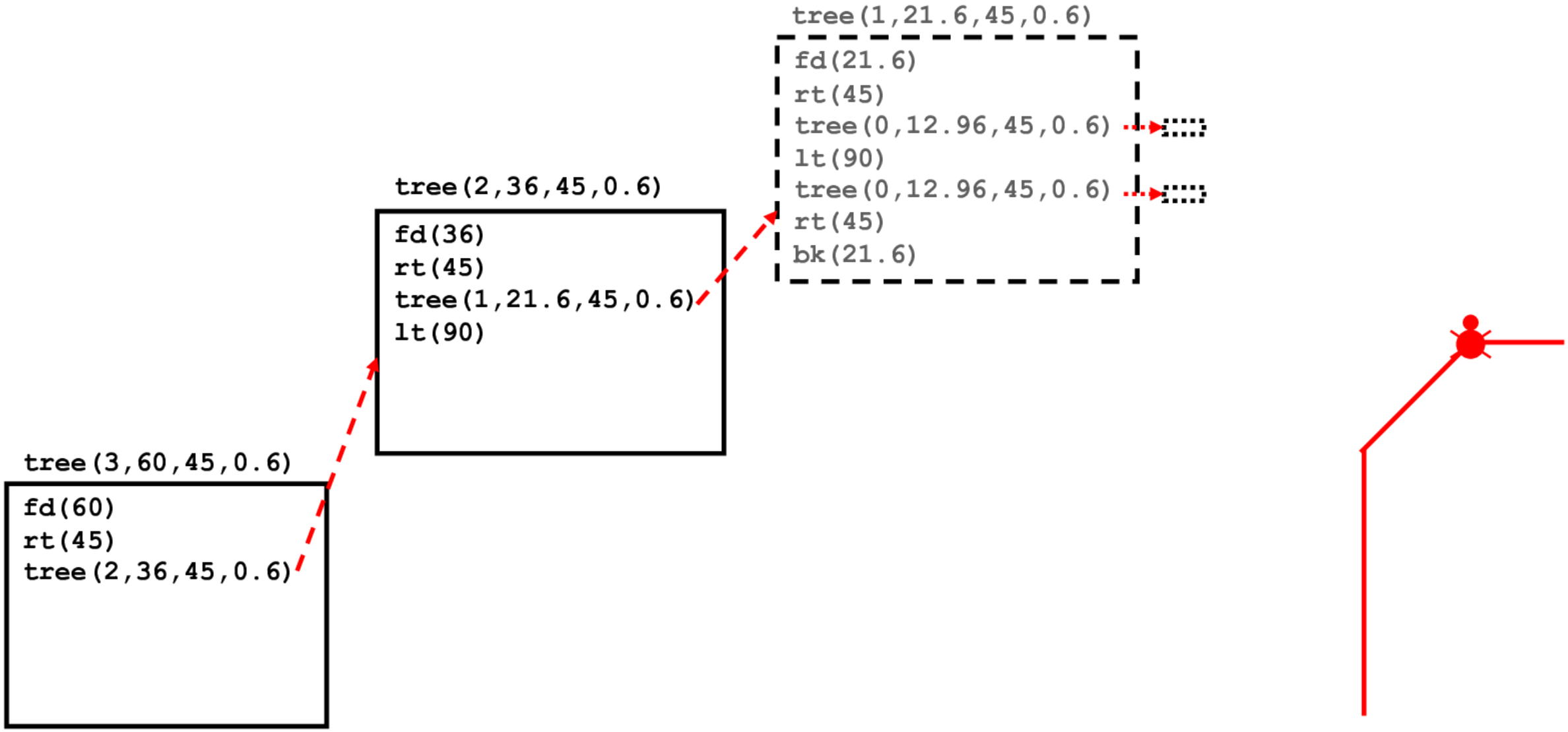
```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
```



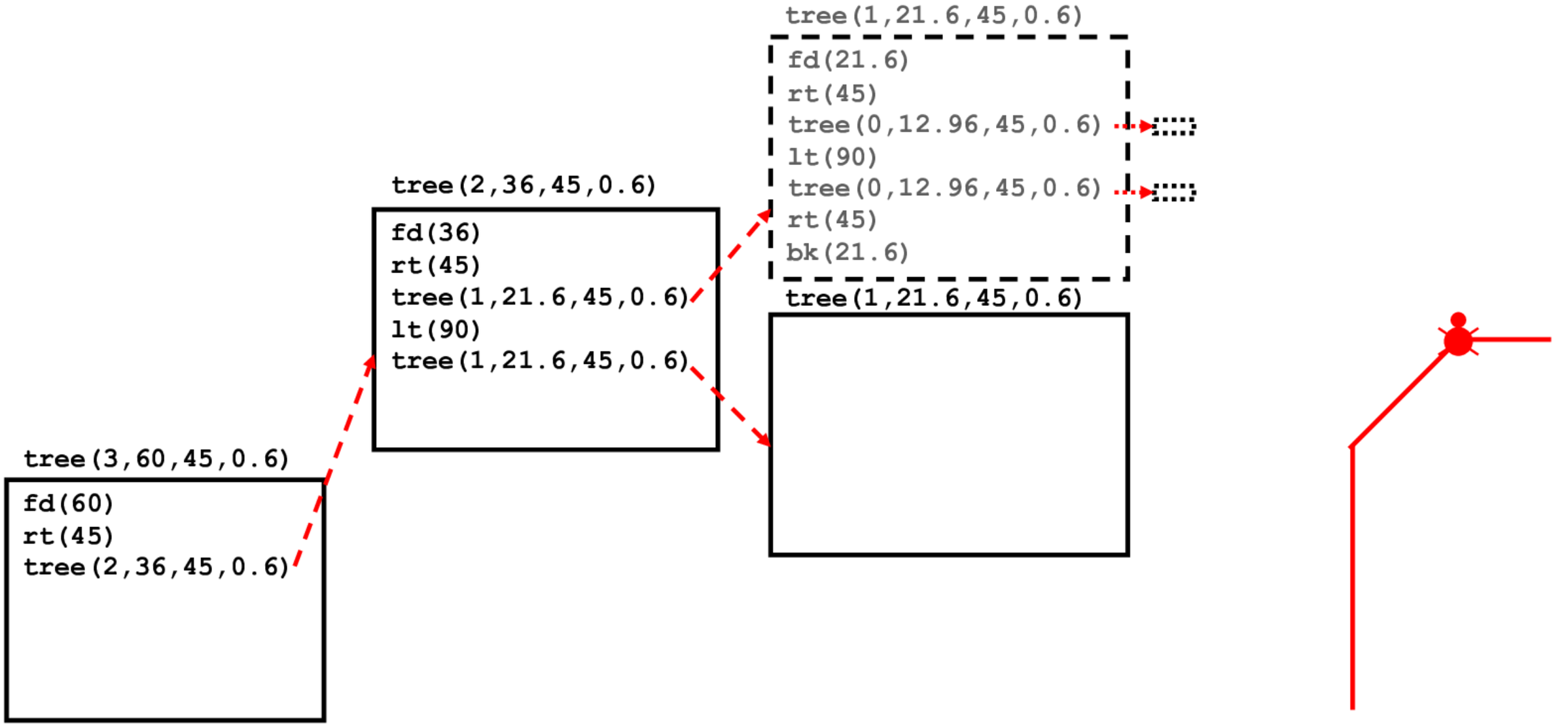
Source: http://cs111.wellesley.edu/spring19

Complete level 1 tree and turn to draw another level 1 tree



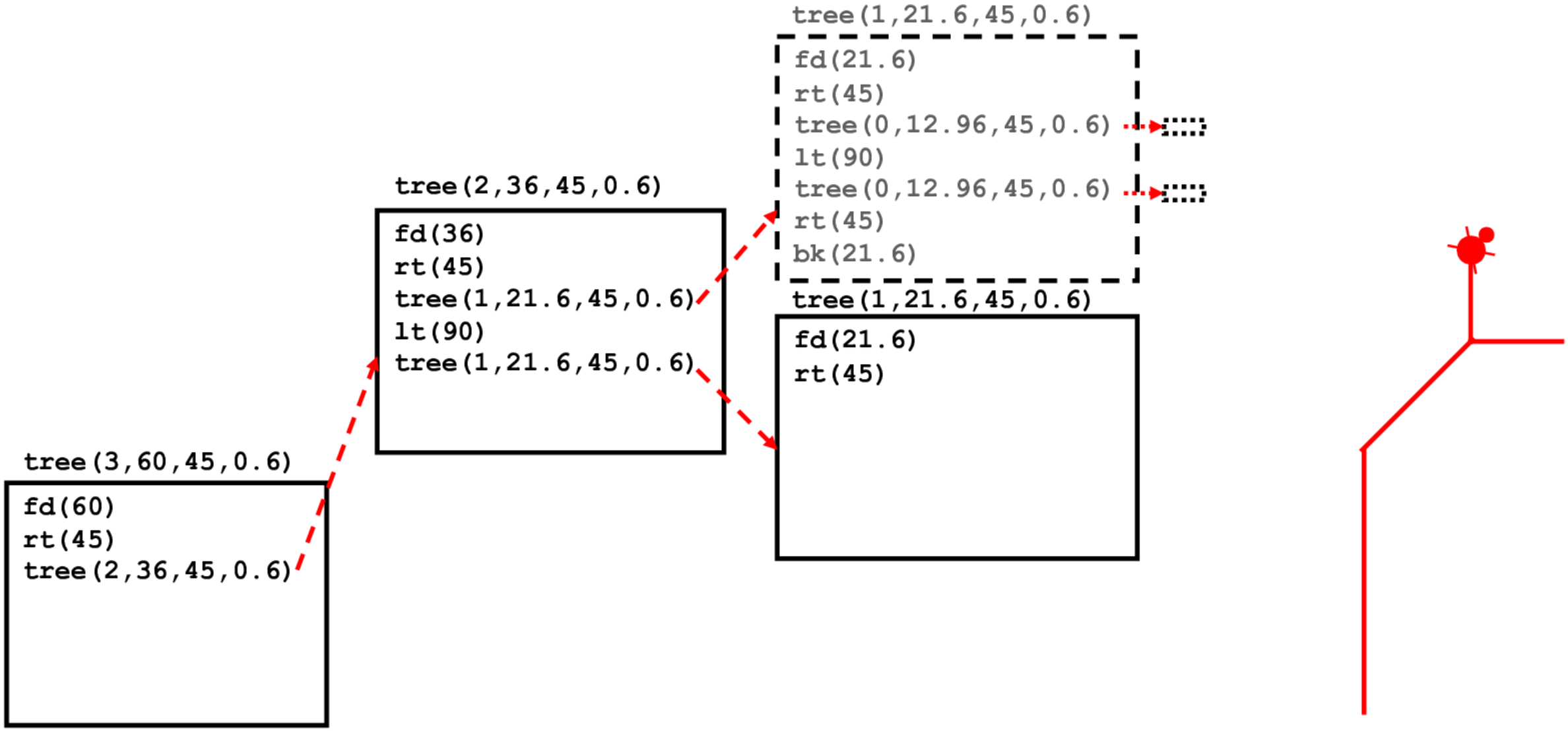
Source: http://cs111.wellesley.edu/spring19

Begin recursive invocation to draw level 1 tree



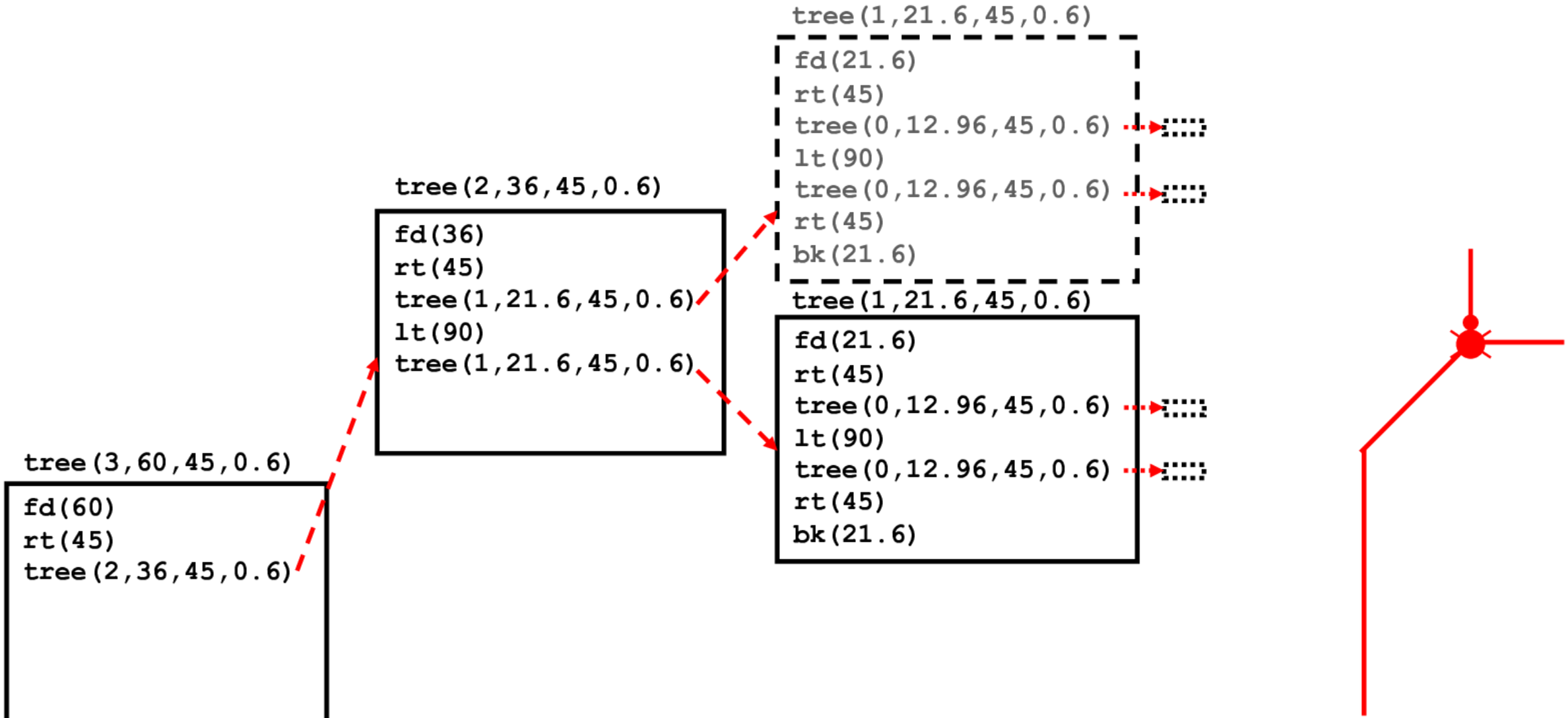
Source: http://cs111.wellesley.edu/spring19

Draw trunk and turn to draw level 0 tree



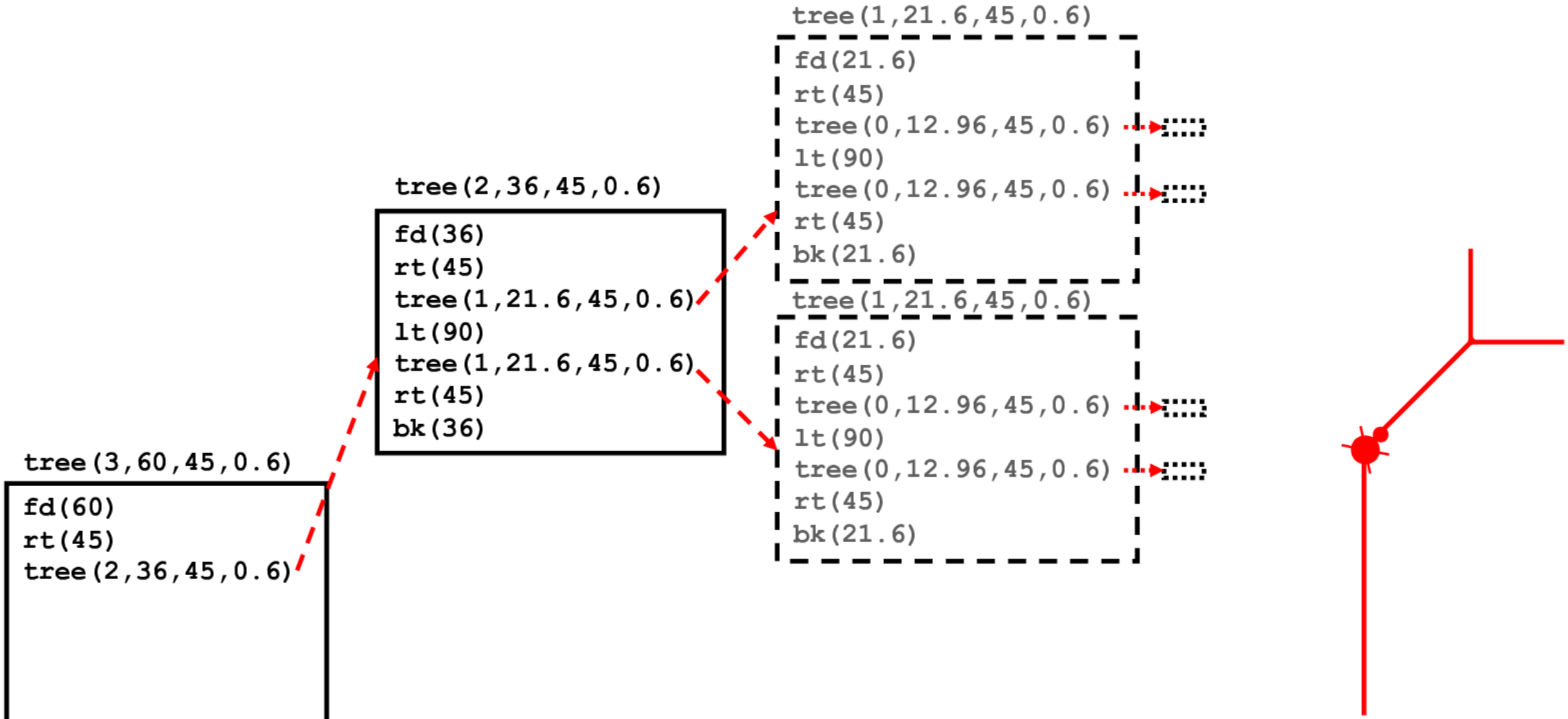
Source: http://cs111.wellesley.edu/spring19

Complete two level 0 trees and return to starting position of level 1 tree



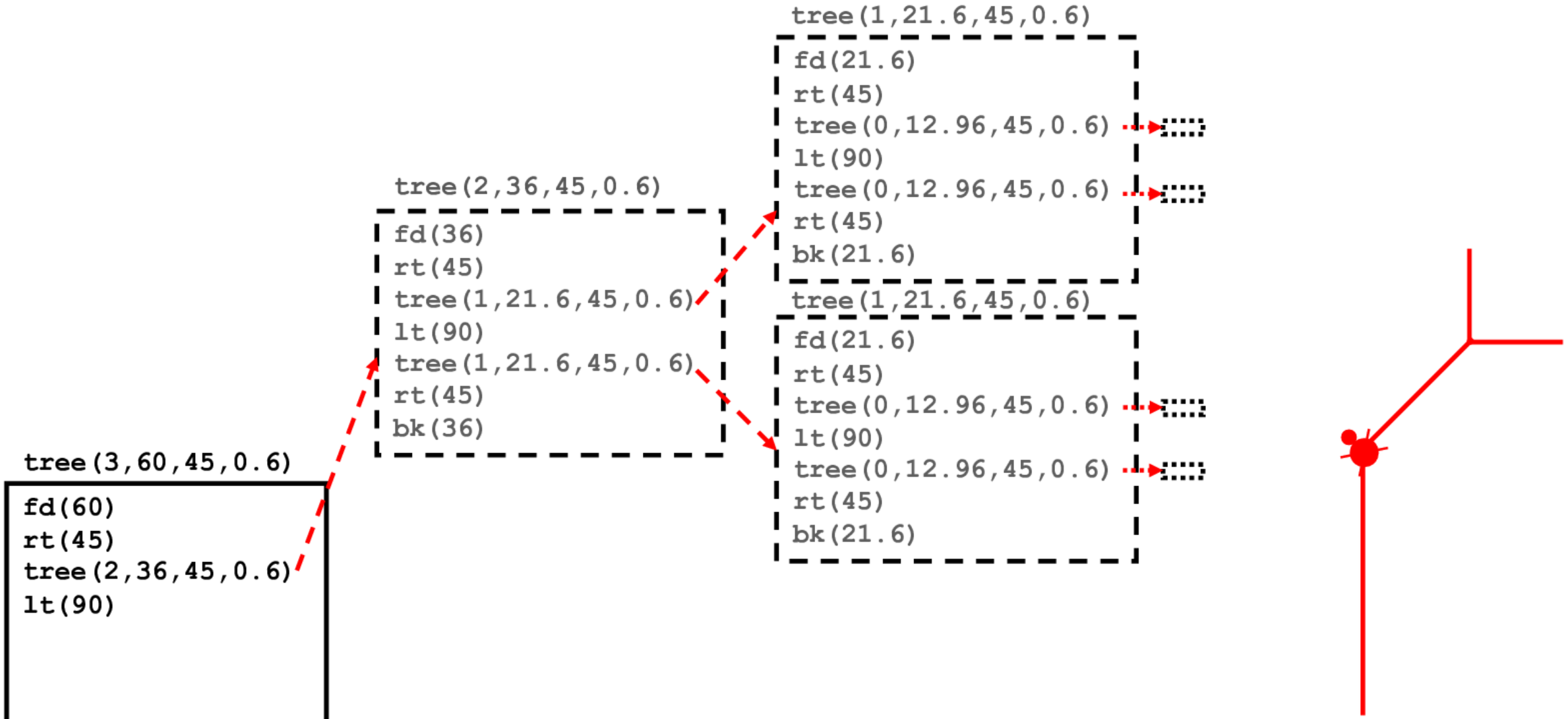
Source: http://cs111.wellesley.edu/spring19

Complete level 1 tree and return to starting position of level 2 tree



Source: http://cs111.wellesley.edu/spring19

Complete level 2 tree and turn to draw another level 2 tree



Source: http://cs111.wellesley.edu/spring19

Draw trunk and turn to draw level 1 tree

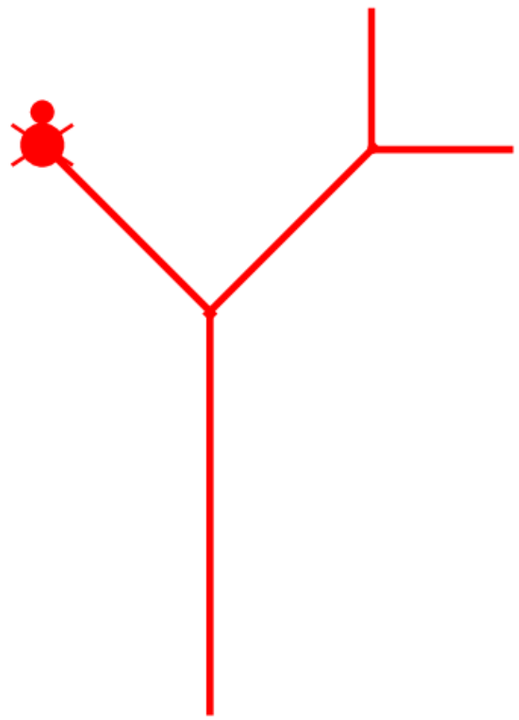
```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
lt(90)
tree(2,36,45,0.6)
```

```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
rt(45)
bk(36)
```

```
tree(2,36,45,0.6)
fd(36)
rt(45)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```



Source: http://cs111.wellesley.edu/spring19

Draw trunk and turn to draw level 0 tree

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
lt(90)
tree(2,36,45,0.6)
```

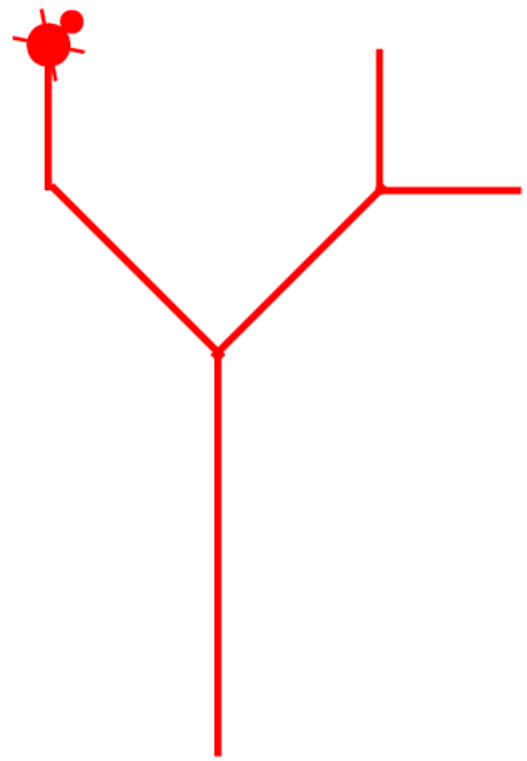
```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
rt(45)
bk(36)
```

```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
```



Source: http://cs111.wellesley.edu/spring19

Complete two level 0 trees and return to starting position of level 1 tree

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
lt(90)
tree(2,36,45,0.6)
```

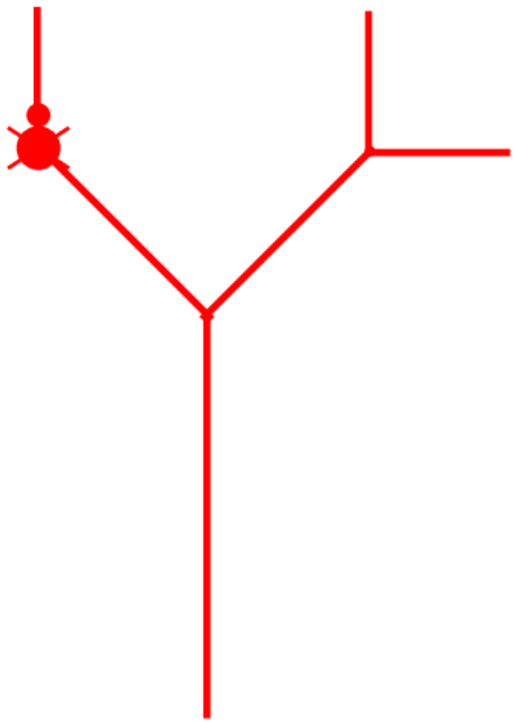
```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
rt(45)
bk(36)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

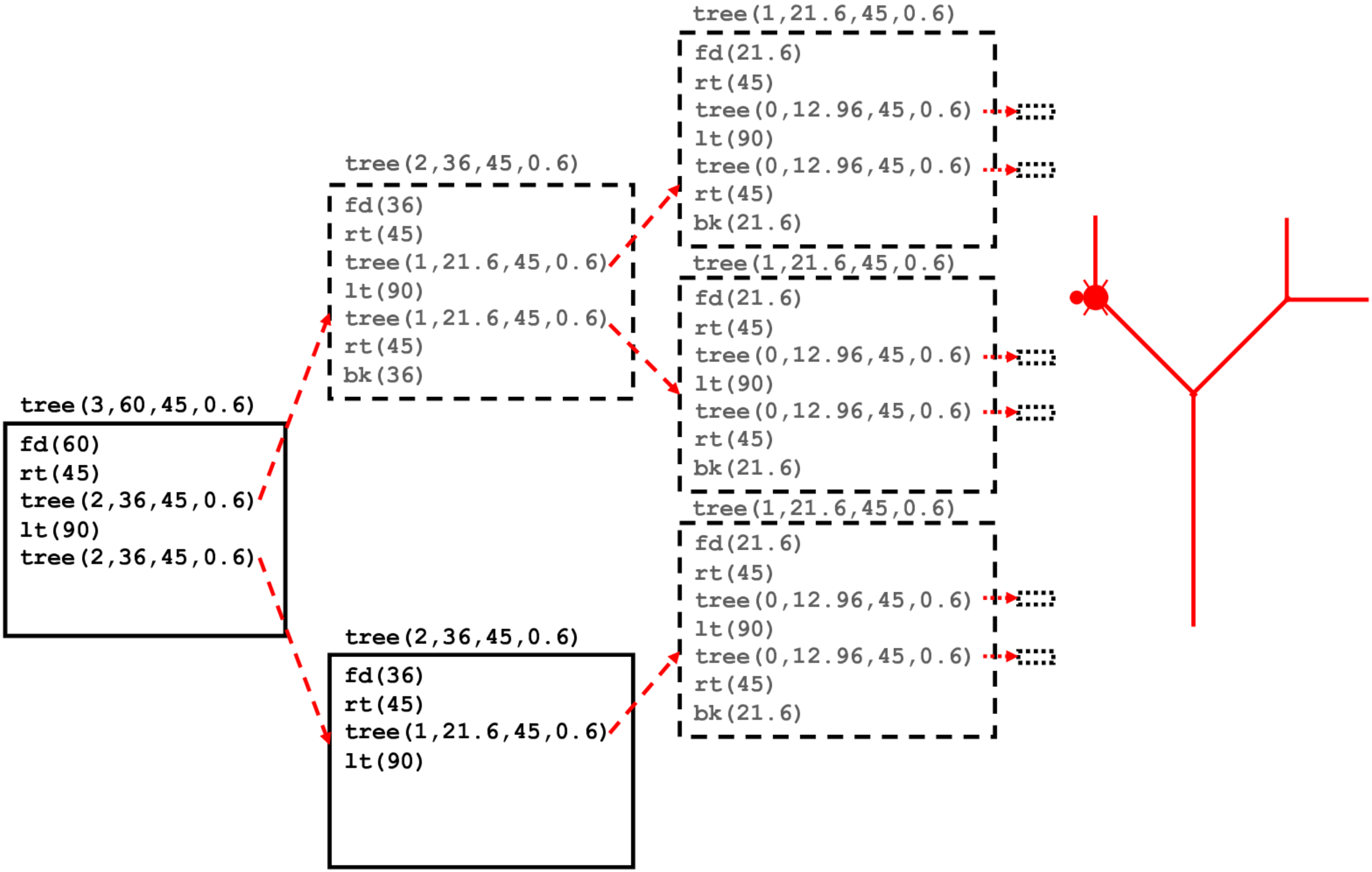
```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```



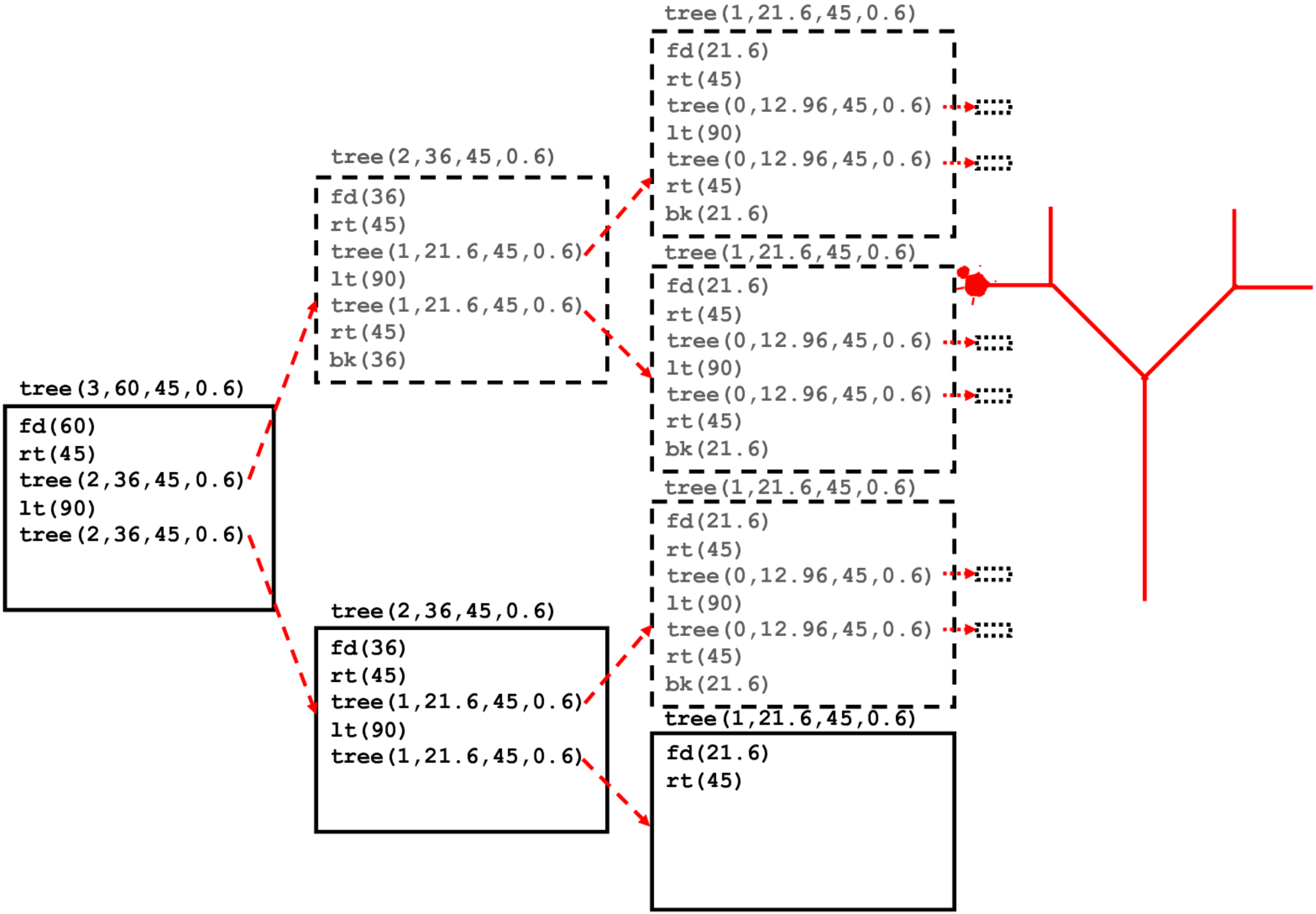
Source: http://cs111.wellesley.edu/spring19

Complete level 1 tree and turn to draw another level 1 tree



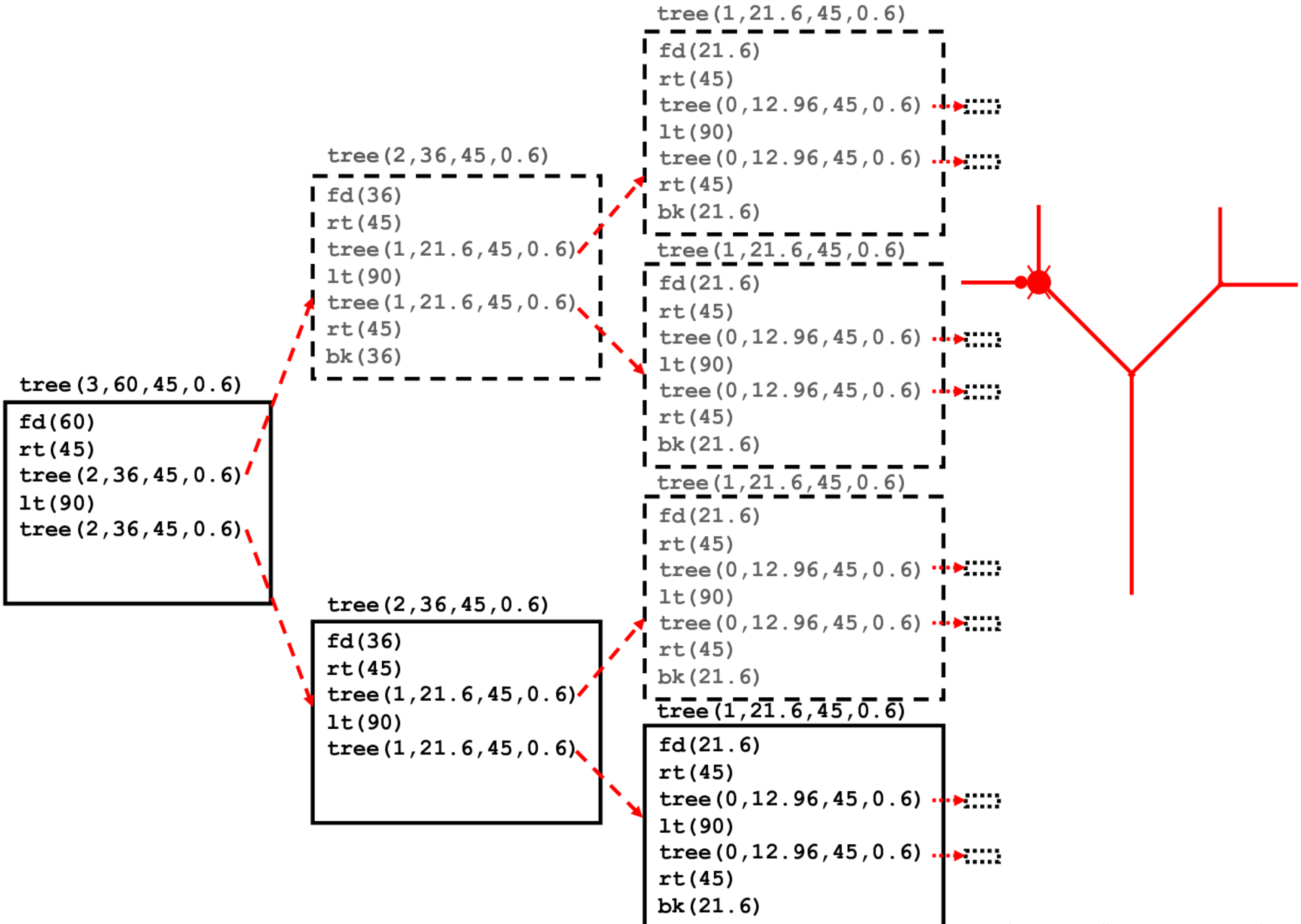
Source: http://cs111.wellesley.edu/spring19

Draw trunk and turn to draw level 0 tree



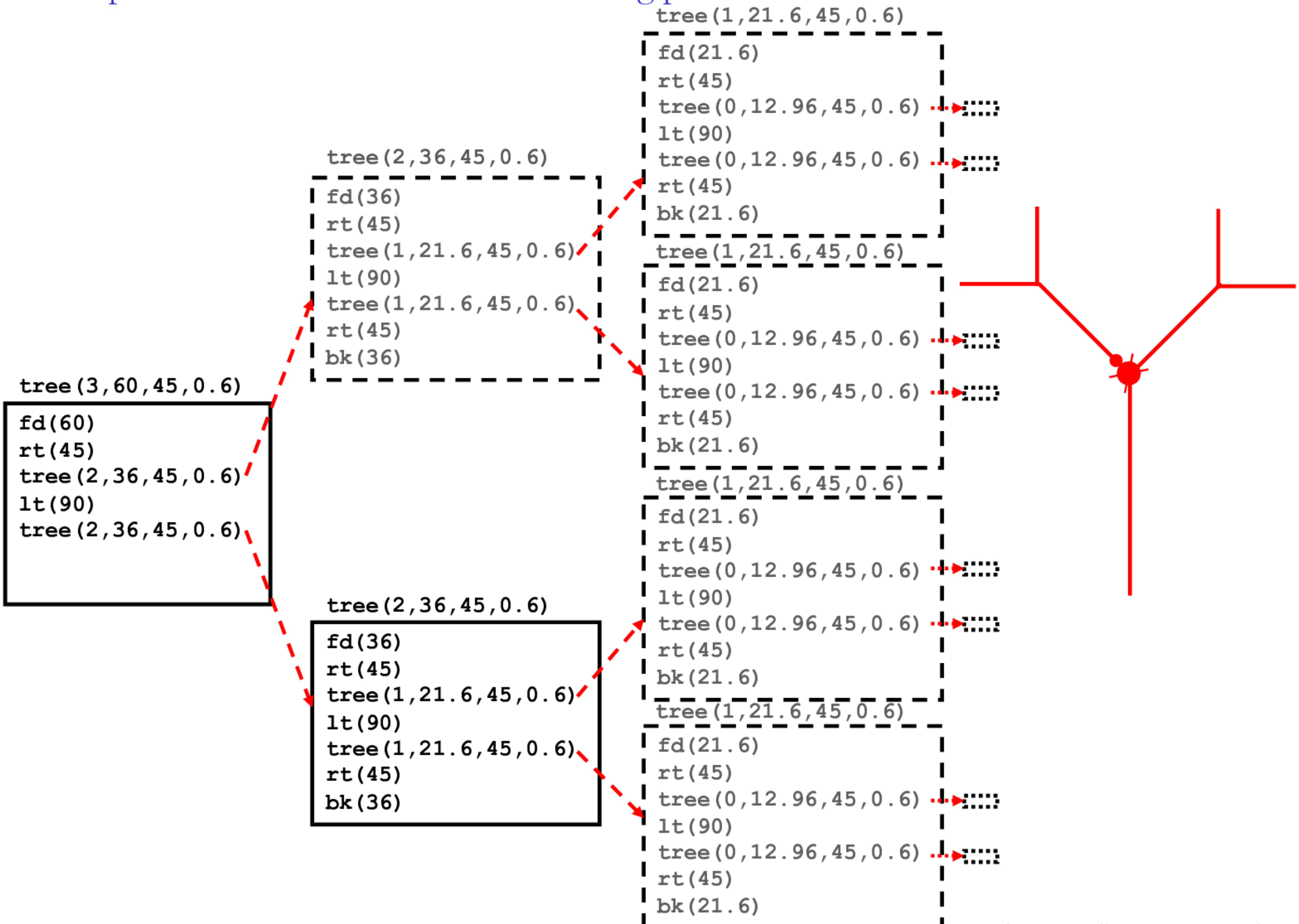
Source: http://cs111.wellesley.edu/spring19

Complete two level 0 trees and return to starting position of level 1 tree



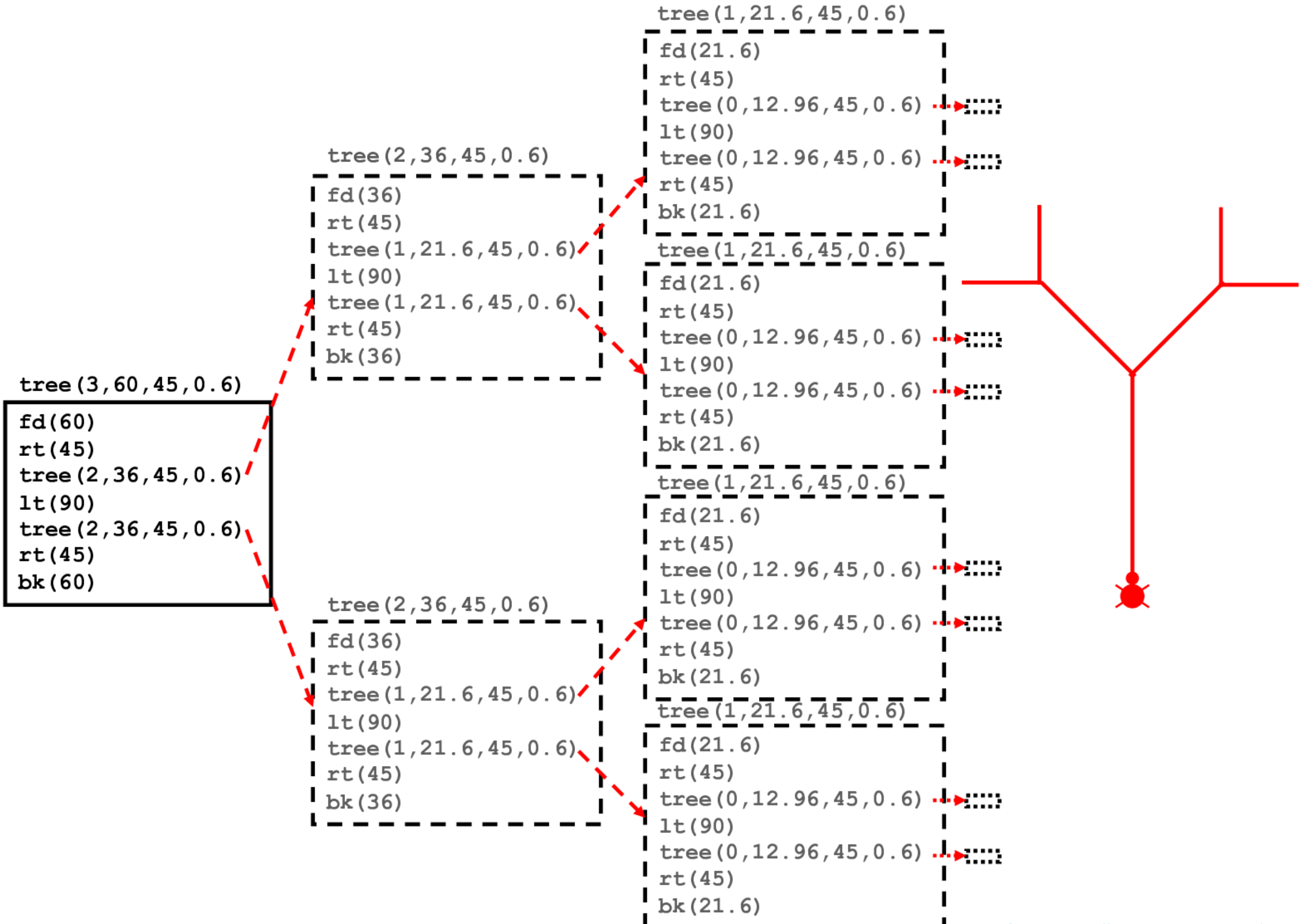
Source: http://cs111.wellesley.edu/spring19

Complete level 1 tree and return to starting position of level 2 tree



Source: http://cs111.wellesley.edu/spring19

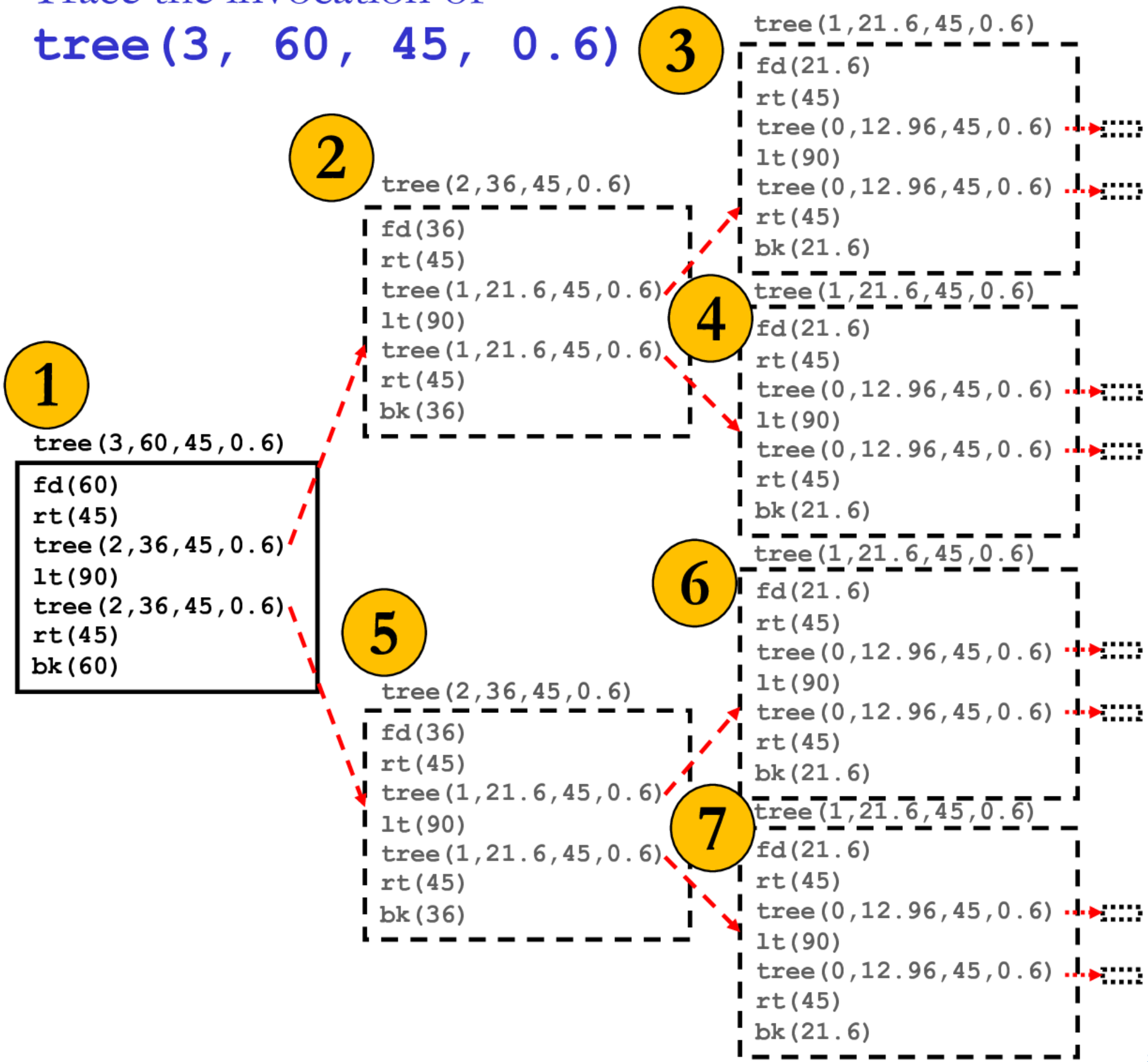
Complete level 2 tree and return to starting position of level 3 tree



Source: http://cs111.wellesley.edu/spring19

Trace the invocation of

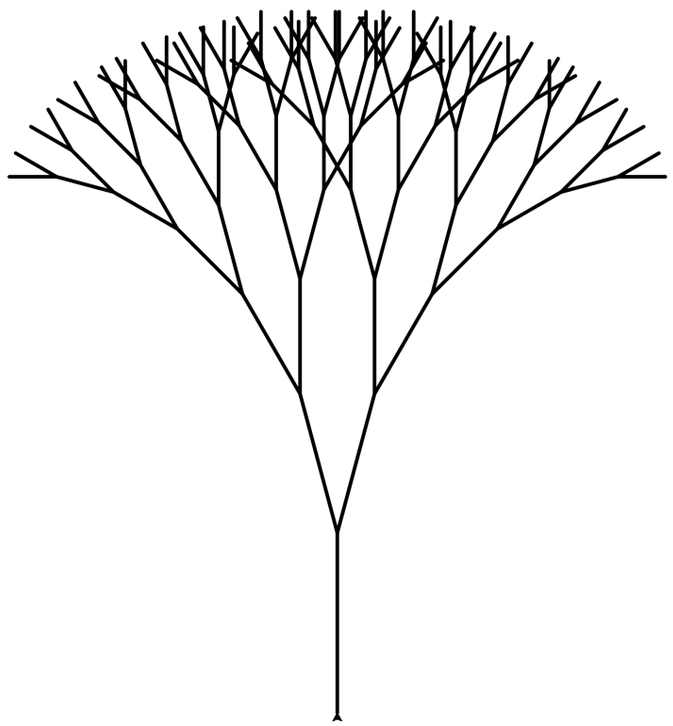
`tree(3, 60, 45, 0.6)`

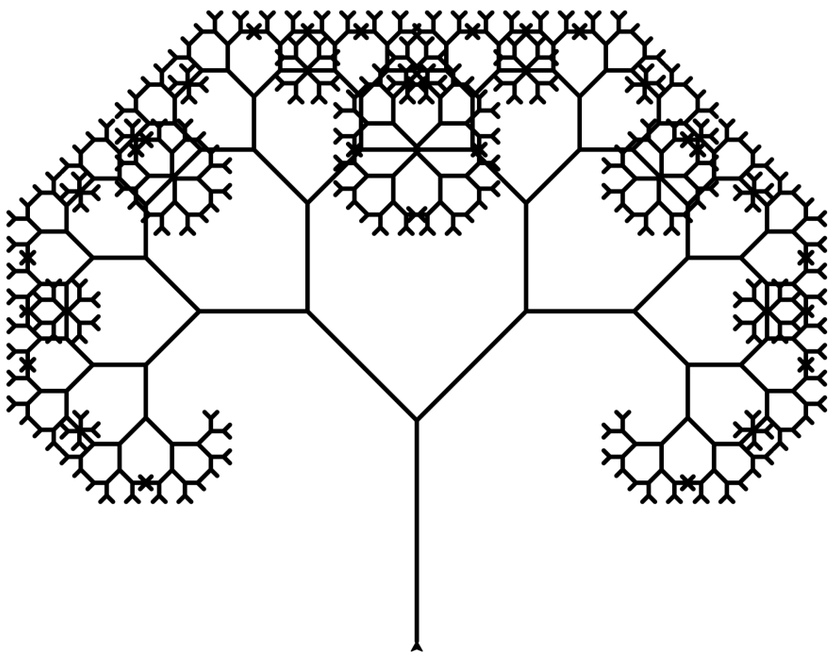


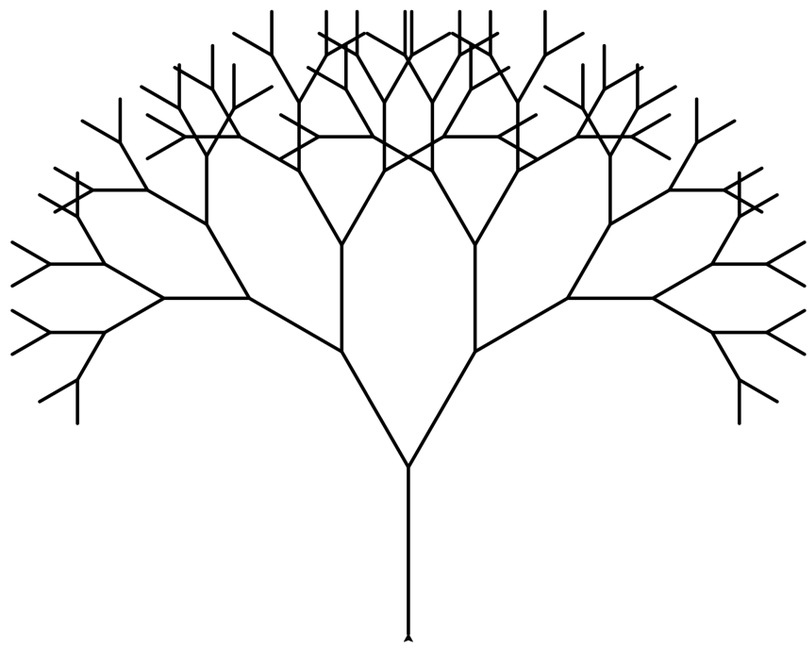
Source: http://cs111.wellesley.edu/spring19

Fruitful Recursion:

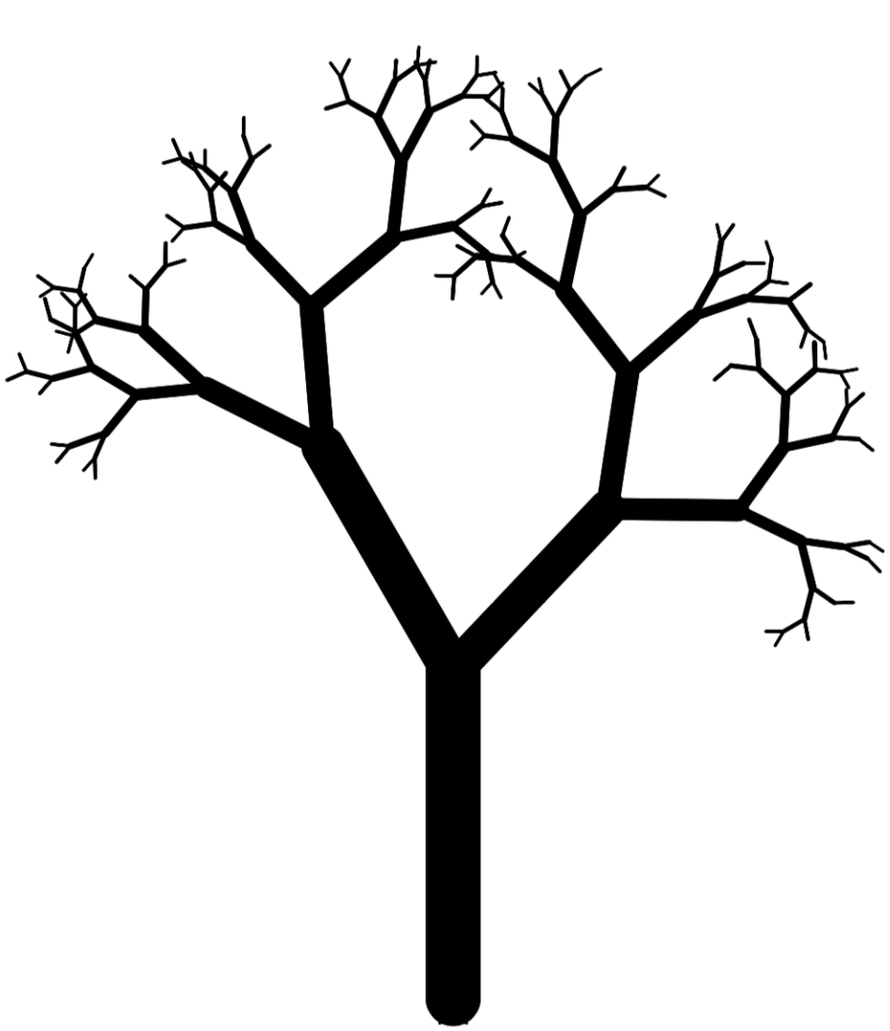
Branch Count

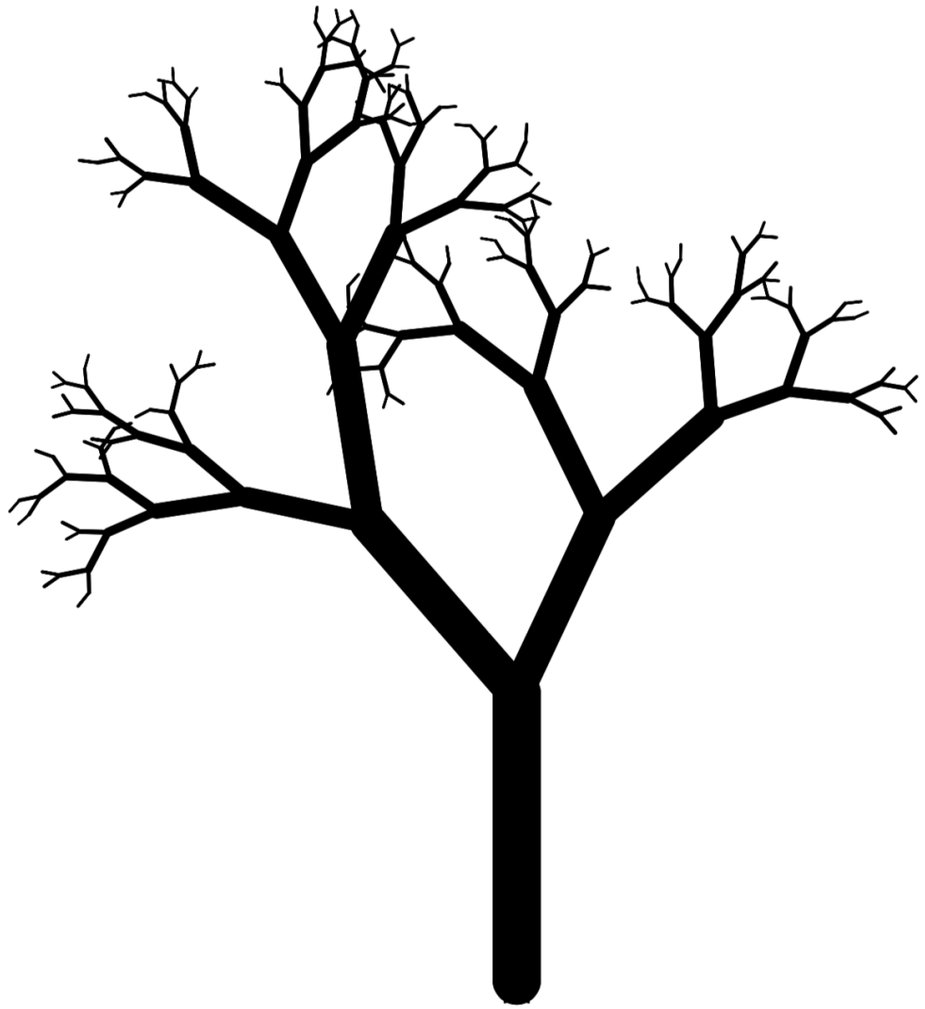




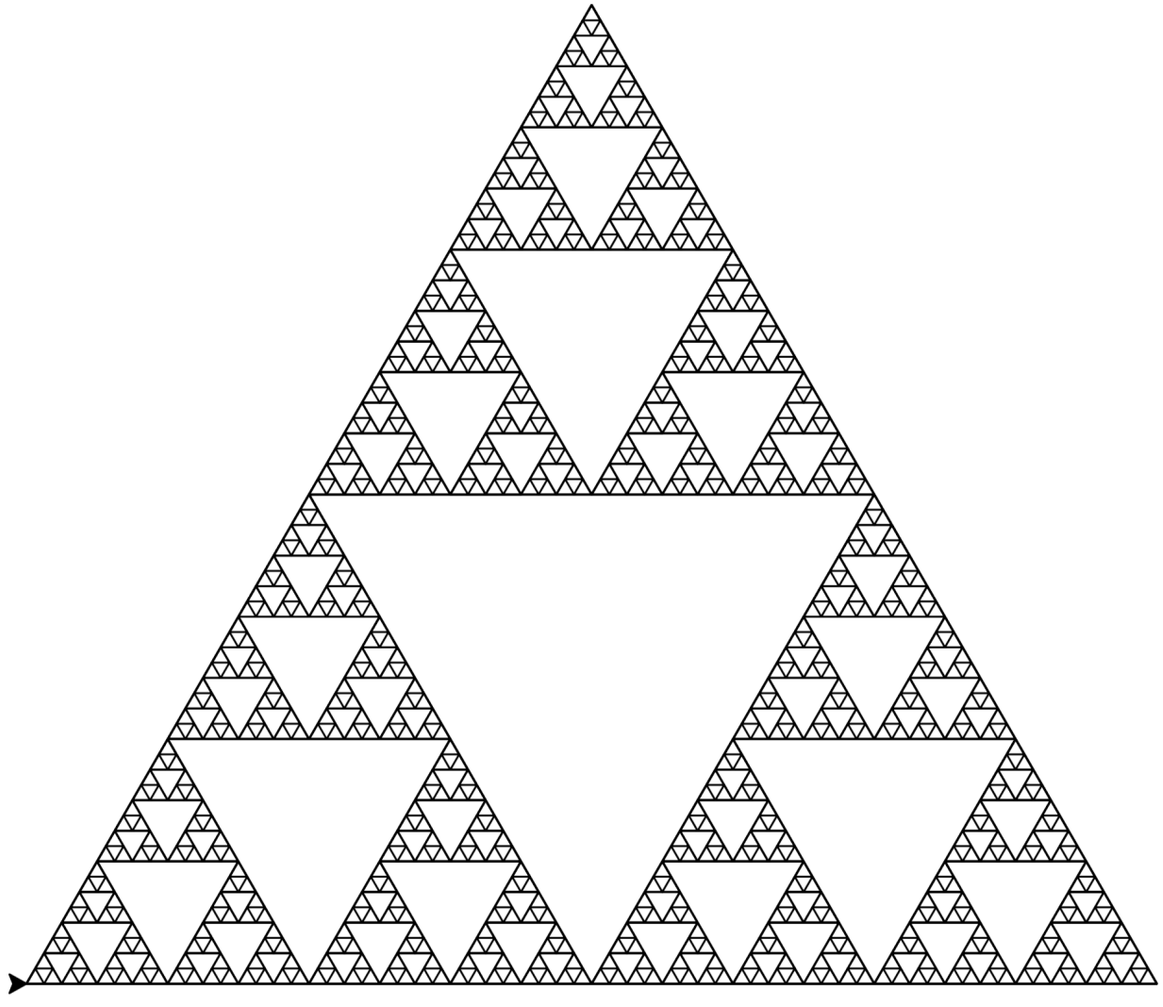


Random Trees





Sierpinski Triangle



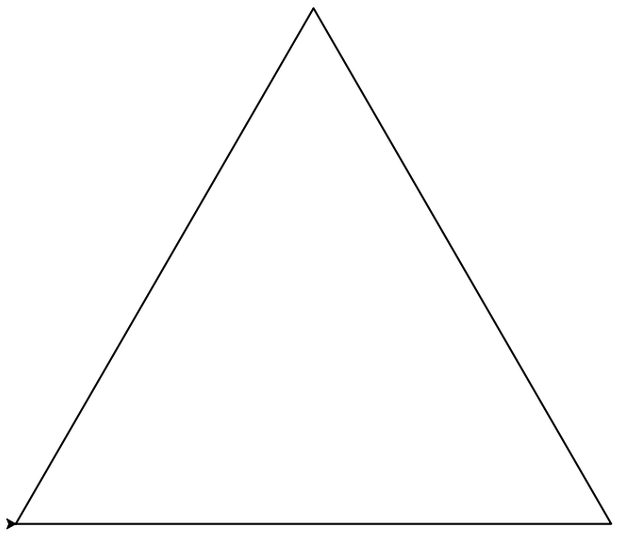
Sierpinski Triangle

`sierpinski(sideLen, level)`

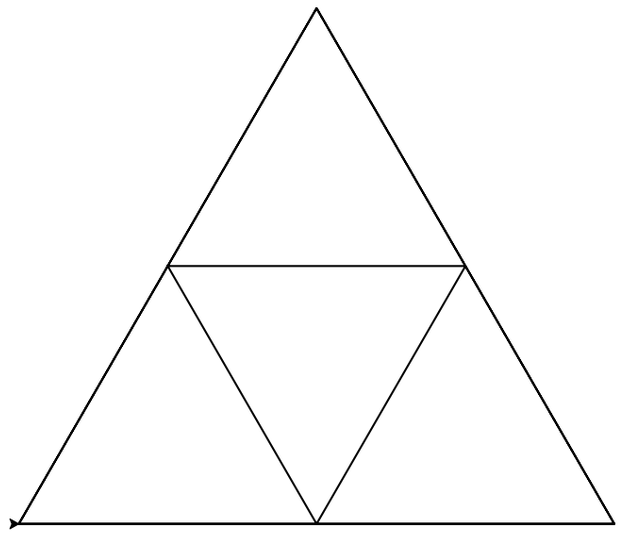
- `sideLen`: length of the outermost triangle
- `level`: determines # of subpatterns:
    - level = 0 nothing is drawn
    - level = 1 only a single triangle (no sub patterns)
    - level = ℓ has level $\ell - 1$ sierpinski triangles as its subpatterns

sierpinski(sideLen, level)

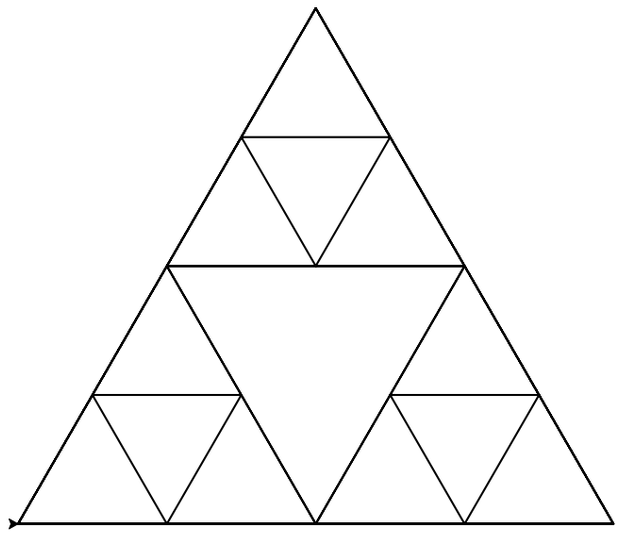
sierpinski(600, 1)



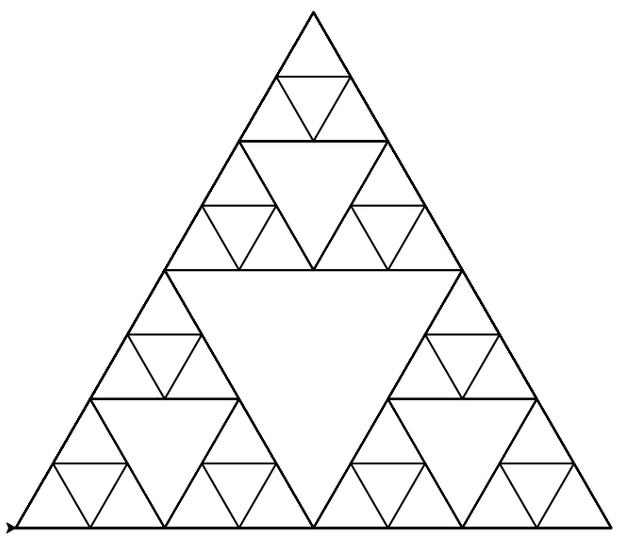
sierpinski(600, 2)



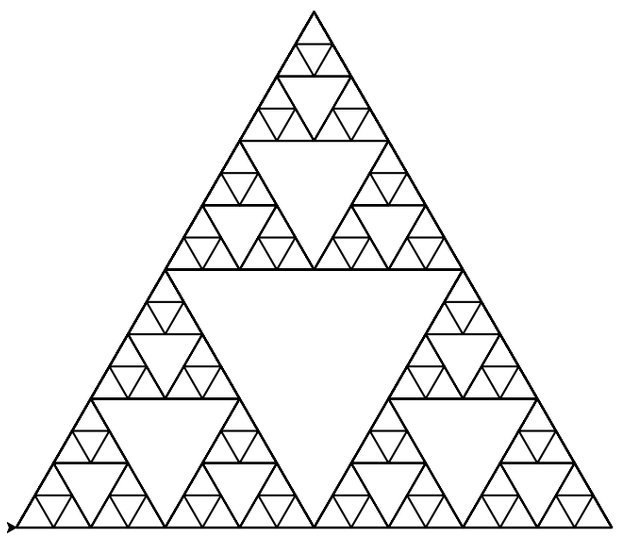
sierpinski(600, 3)



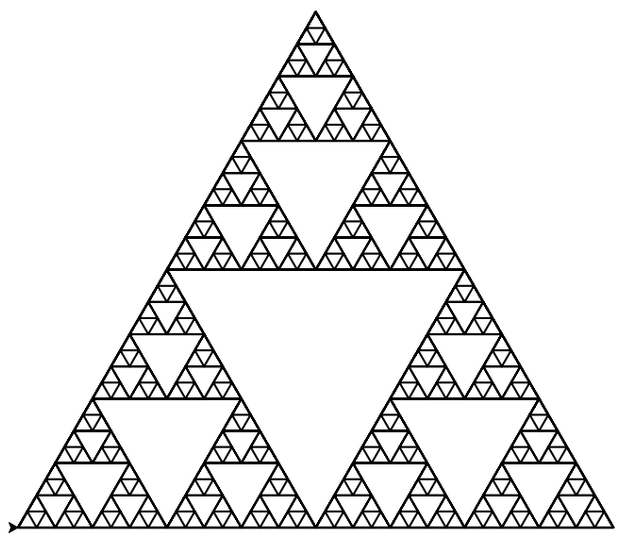
sierpinski(600, 4)



sierpinski(600, 5)



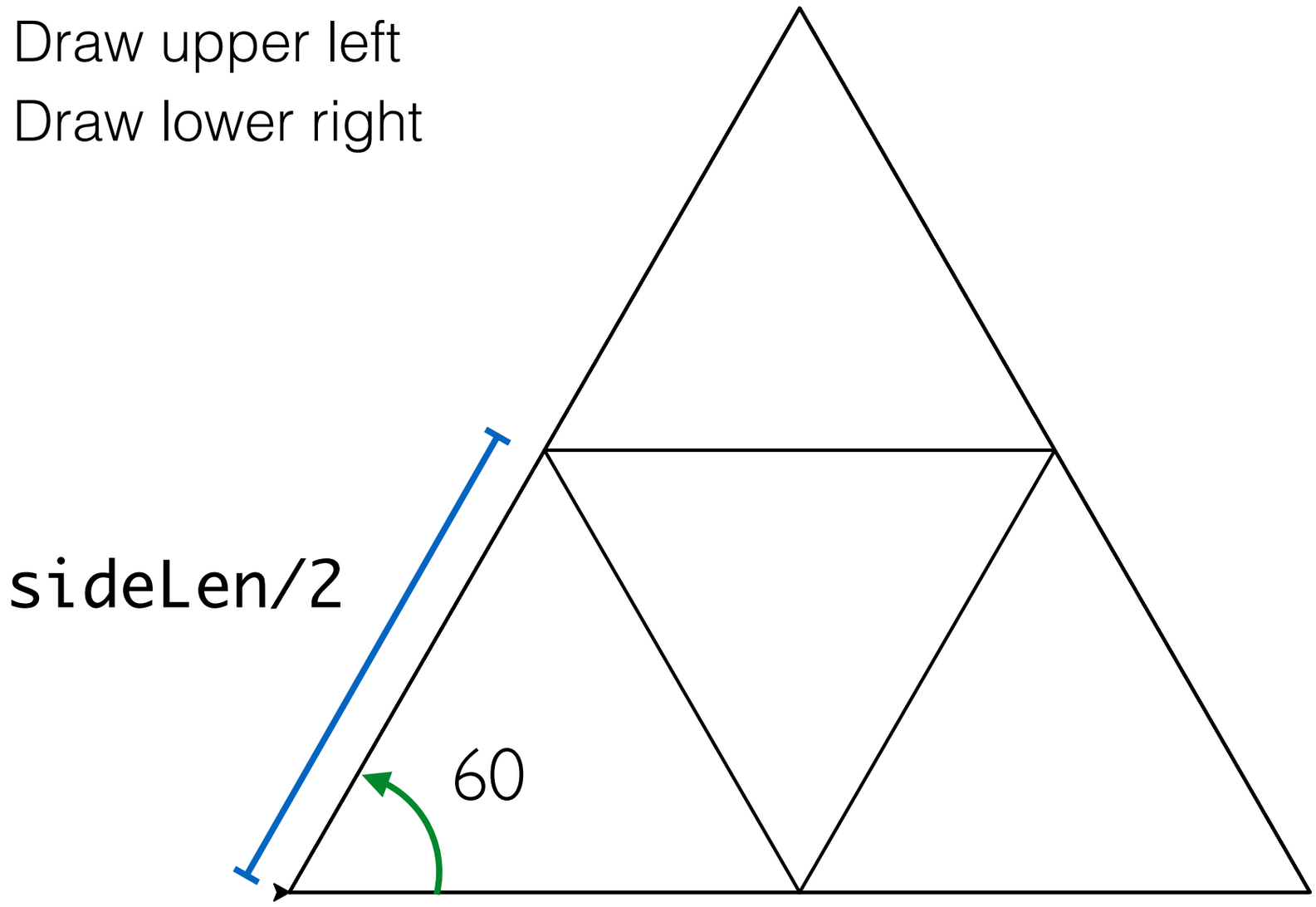
sierpinski(600, 6)

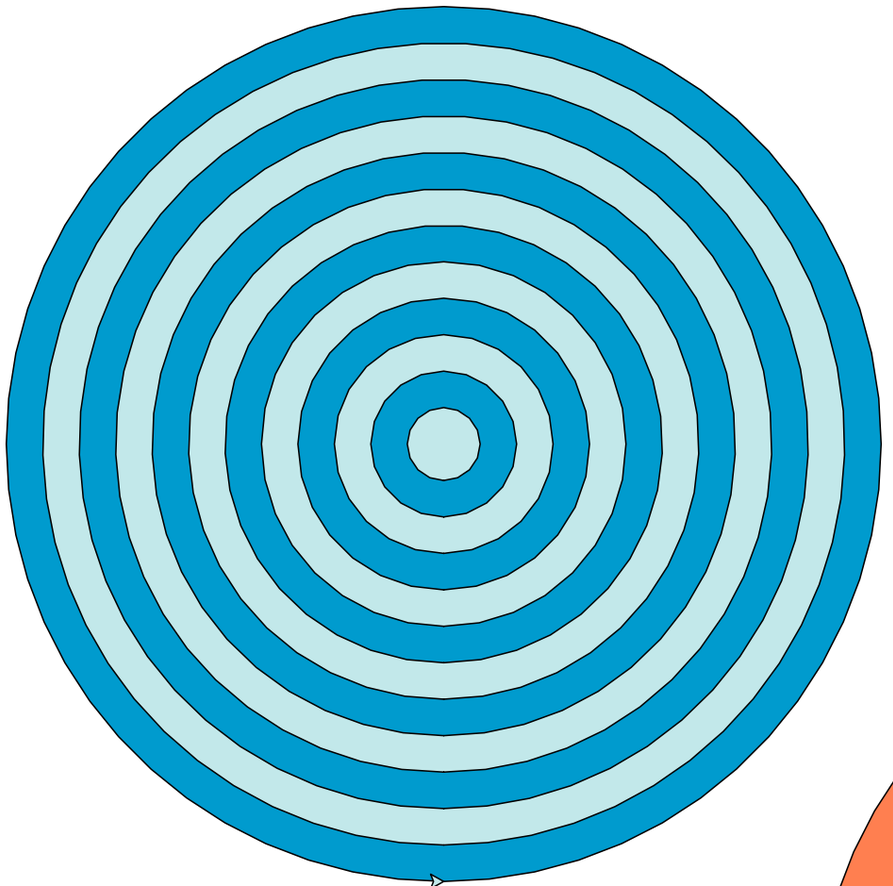


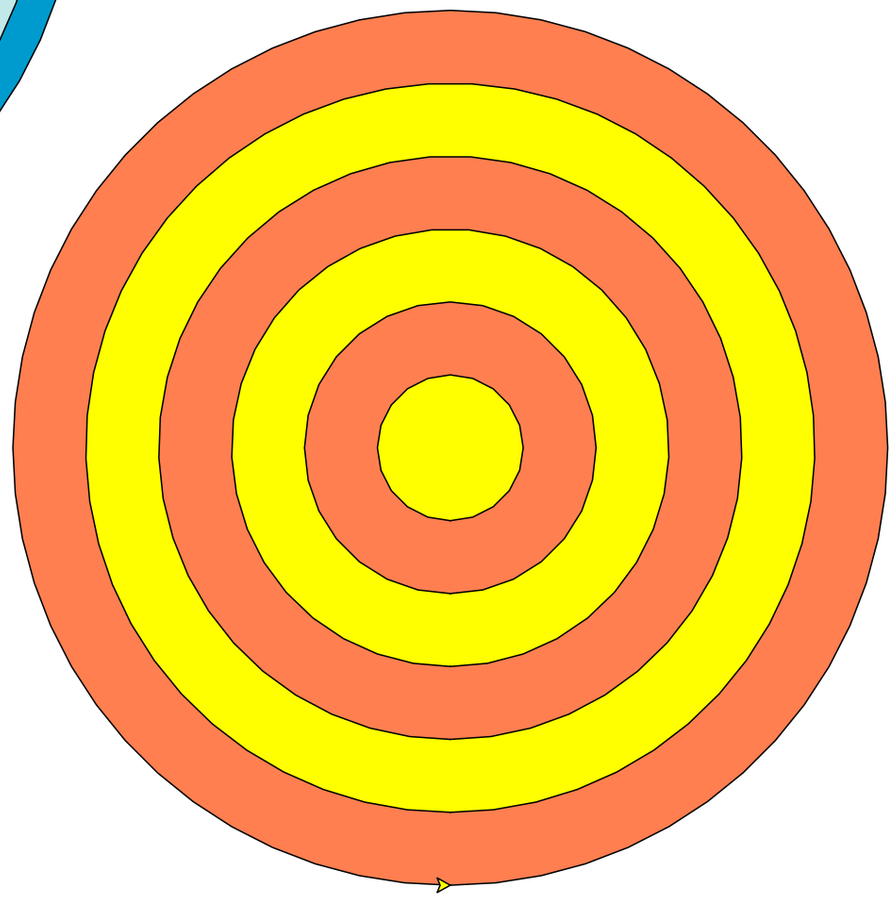
```
def drawTriangle(sideLen):
    """Draws triangle with sides
    of length sideLen starting
    from one end point"""
    pd()
    fd(sideLen)
    lt(120)
    fd(sideLen)
    lt(120)
    fd(sideLen)
    lt(120)
    pu()
```

`sierpinski(sideLen, level)`

- First draw outer big triangle
- Then recursively
  - Draw upper triangle
  - Draw upper left
  - Draw lower right







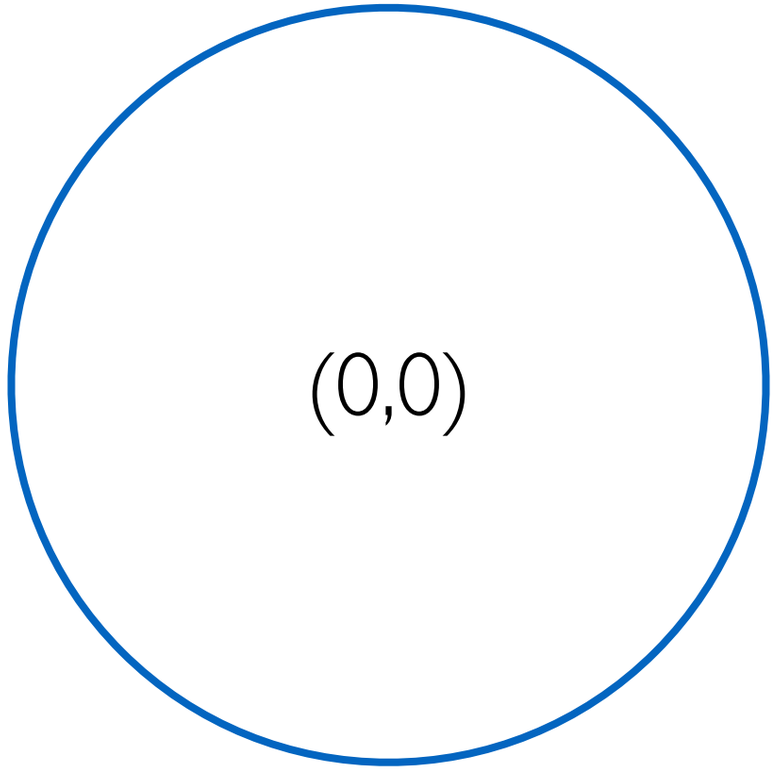
Concentric Circles

Concentric Circles

```
concentricCirc(radius, thickness, color1, color2)
```

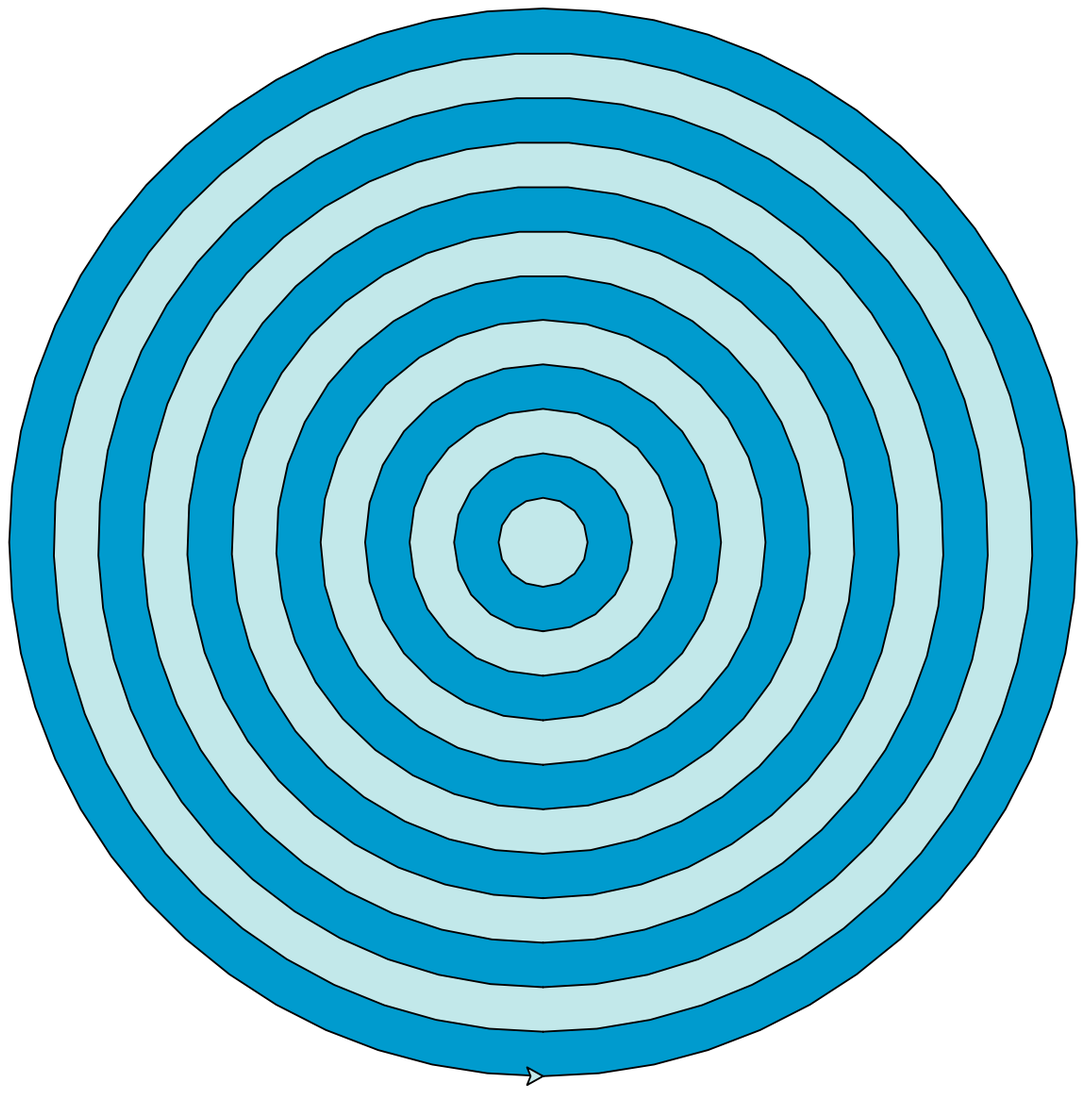
- `radius`: radius of the outermost circle
- `thickness`: thickness of the band between circles
- `color1`: color of the outermost circle
- `color2`: color that alternates with color1

```
def drawDisc(radius, color):
    """Draws a circle of given
radius and color with centre
(0,0) assuming turtle's initial
position is (0, -radius)"""
```



Starting position of turtle (0, -radius)

```
concentricCirc(radius,
thickness, color1, color2)
```



Starting position of turtle

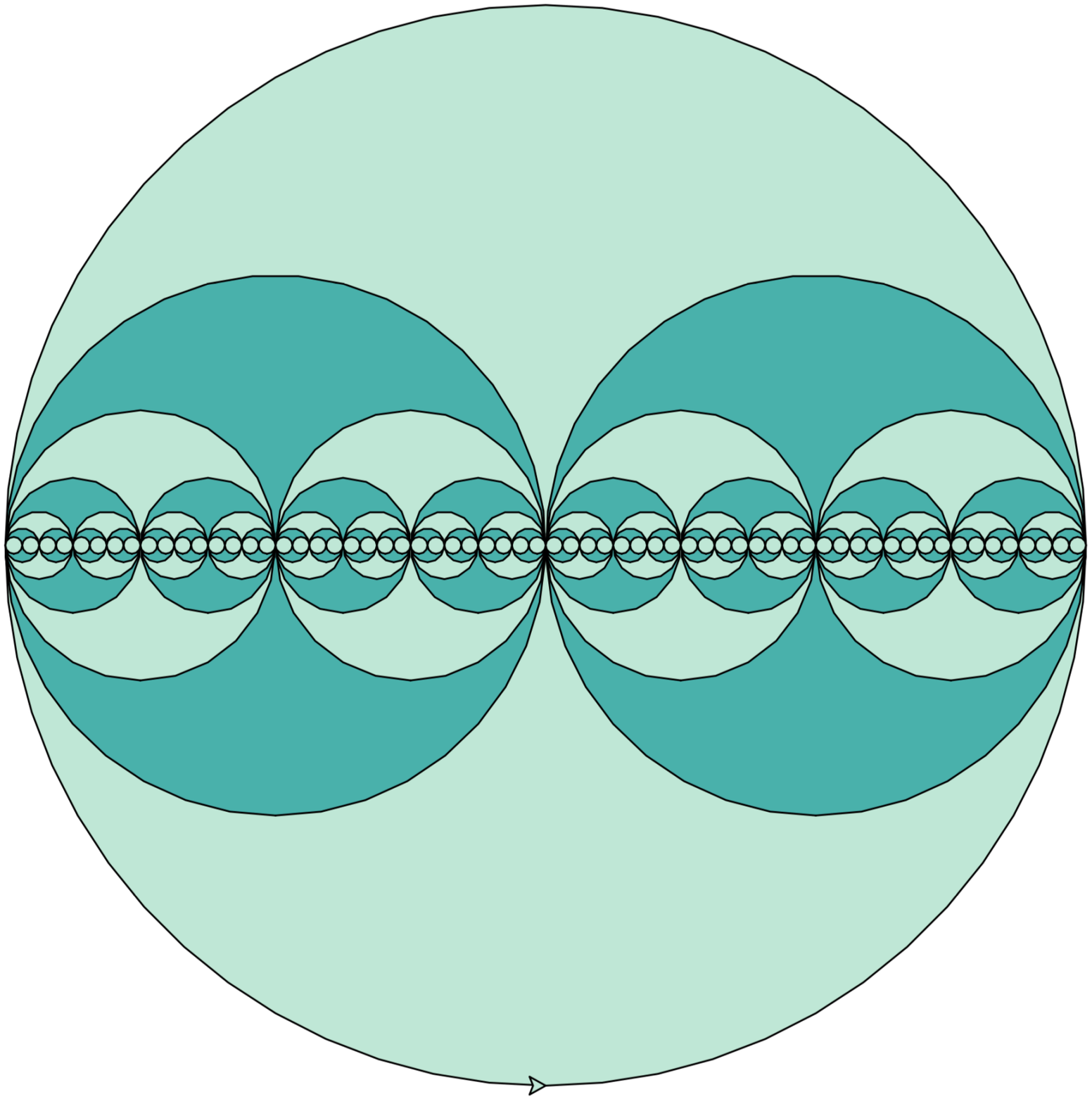
Fruitful version

`concentricCirc(radius, thickness, color1, color2)`

- `radius`: radius of the outermost circle
- `thickness`: thickness of the band between circles
- `color1`: color of the outermost circle
- `color2`: color that alternates with color1

Must return tuple of values:

- first item is # of circles of `color1`
- second item is # of circles of `color 2`

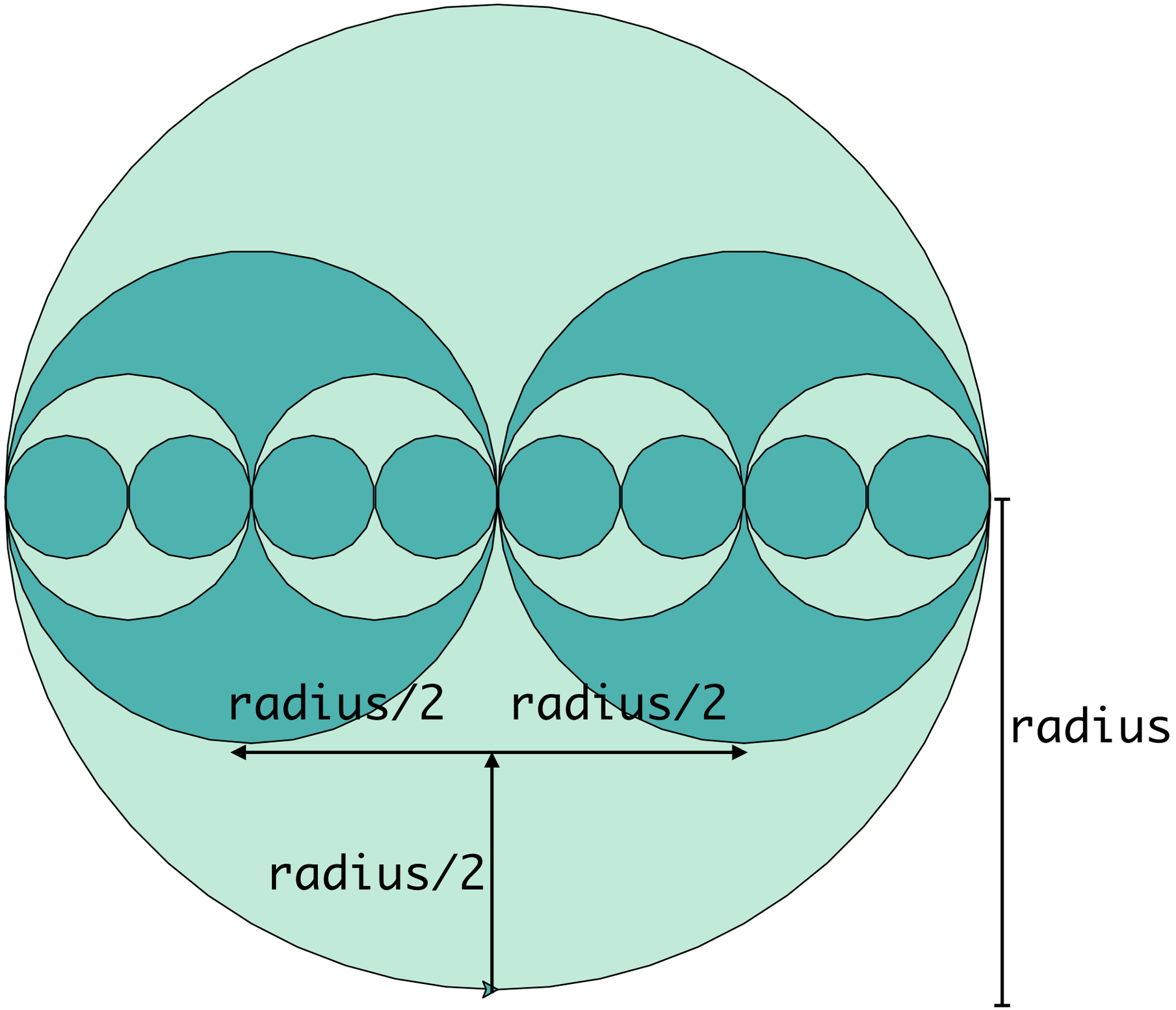


Nested Circles

Nested Circles

```
nestedCircles(radius, minRadius, color1, color2)
```

- `radius`: radius of the outermost circle
- `minRadius`: minimum radius of any circle
- `color1`: color of the outermost circle
- `color2`: color that alternates with `color1`



Starting position of turtle

```
nestedCircles(300, 37.5)
```

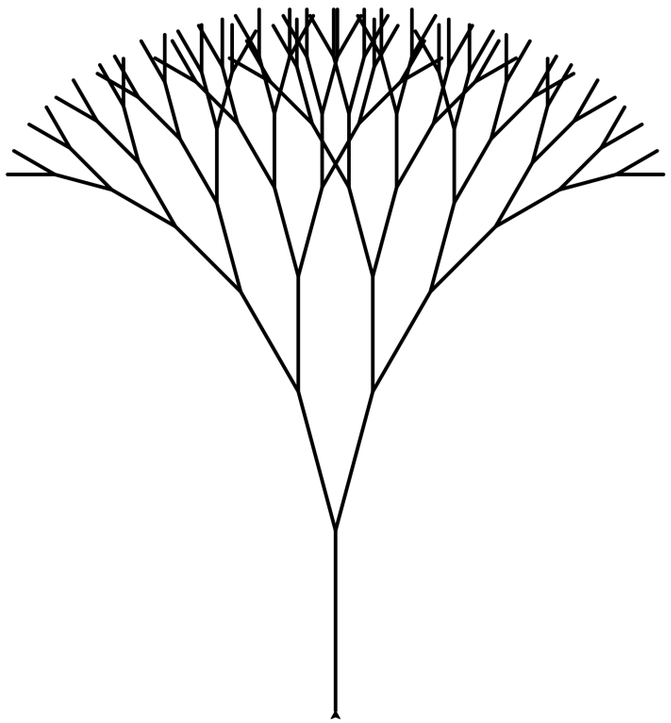
Fruitful version

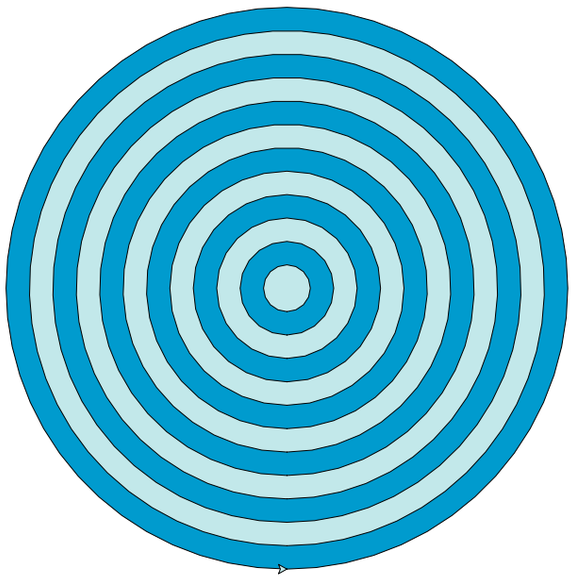
`nestedCircles(radius, minRadius, color1, color2)`

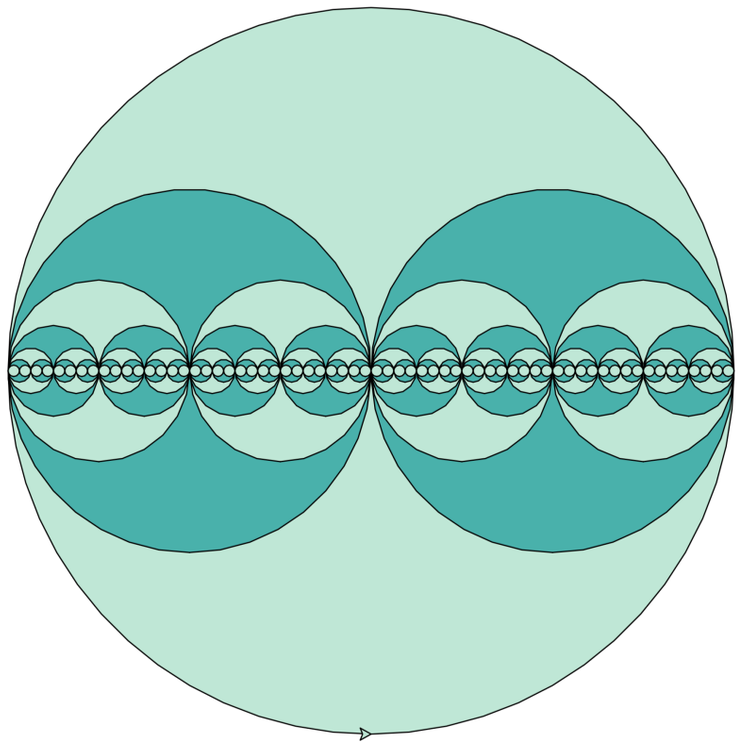
- `radius`: radius of the outermost circle
- `minRadius`: minimum radius of any circle
- `color1`: color of the outermost circle
- `color2`: color that alternates with `color1`

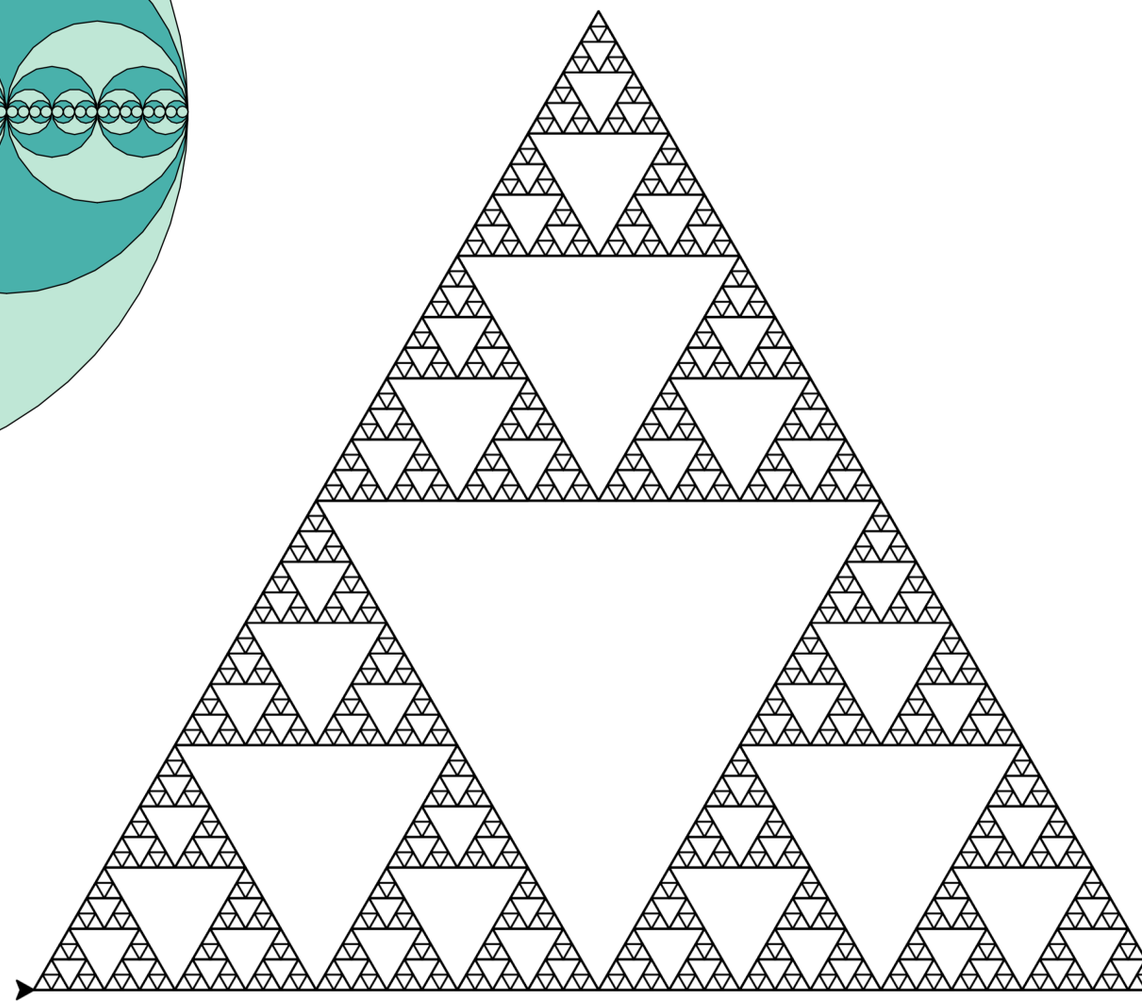
Must return tuple of values:

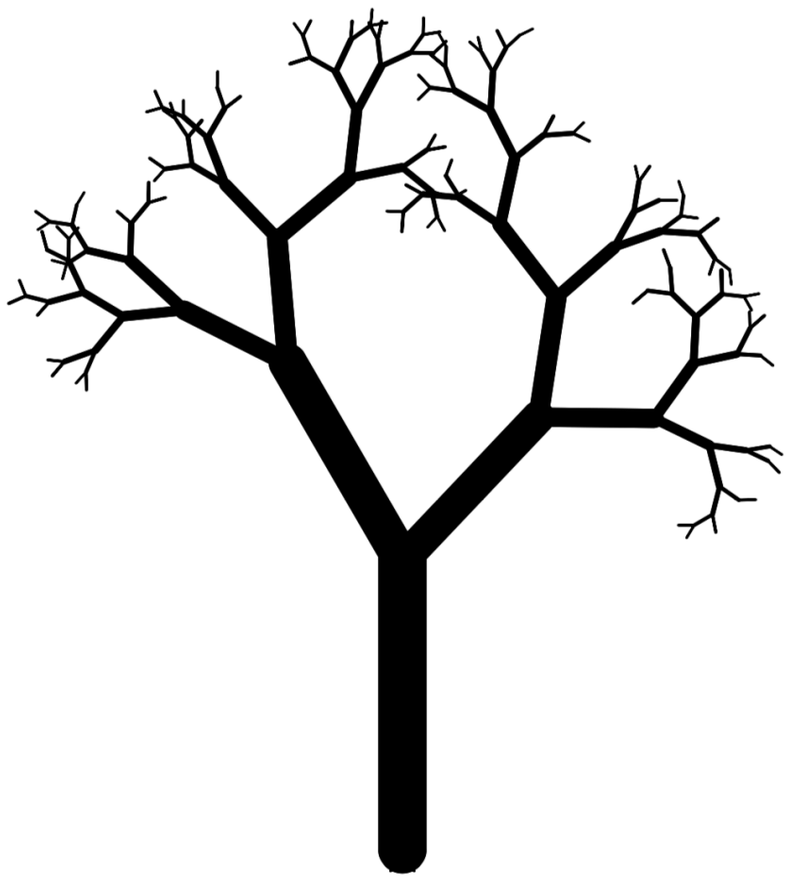
- first item is # of circles of `color1`
- second item is # of circles of `color 2`











Acknowledgments

These slides have been adapted from:

- http://cs111.wellesley.edu/spring19 and
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/