Computer Programming Fundamentals

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/

QUIZ 3

90 minutes

48 hour window

11am 10/29 - 11am 10/31

Debugging and Recursion

questions?

TODAY: TURTLES & TREES

TAKE 2!

RE-COPY AND PASTE BasicPanel.java FROM CLASS SCHEDULE

REINSTALL TURTLE LIBRARY

FOLLOW LINK FROM CLASS SCHEDULE, DOWNLOAD Turtle.jar

Turtle for Java

A Library by Leah Buechley. The Turtle library provides an implementation of a LOGO Turtle for Java.

Turtle Geometry (see the fabulous book of the same name by Hal Abelson and Andrea diSessa) provides a different way of thinking about geometry. You draw by driving around a "turtle". Programs are written from the point of view of this turtle, which enables you to take an embodied approach to geometry.

LOGO, a turtle-based programming language, was developed by Seymour Papert and a group of collaborators in the late 1960s. It was presented as a novel way to introduce children to computer programming and mathematics. LOGO and Turtle Geometry remain strongly associated with children and education, but are full of beautiful tools and ideas that adult artists and programmers can fruitfully explore.

DOWNLOAD

Download Turtle: Turtle.jar.

Reference

Basics

Turtle ()

forward ()
back ()
right ()
left ()

penUp ()
penDown ()

push ()
pop ()

goToPoint ()

drawPath ()
drawTurtle ()

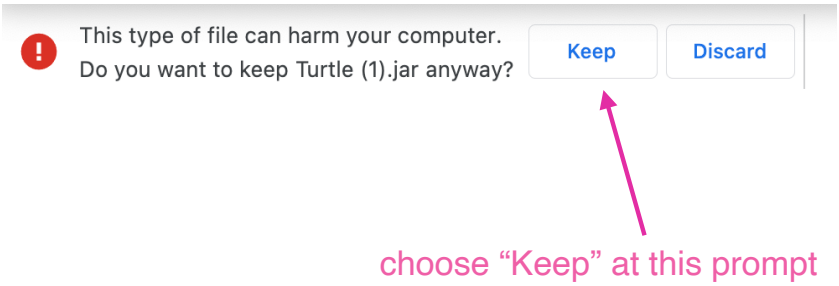
Turtle State

getX ()
getY ()
getHeading ()

setX ()
setY ()
setHeading ()

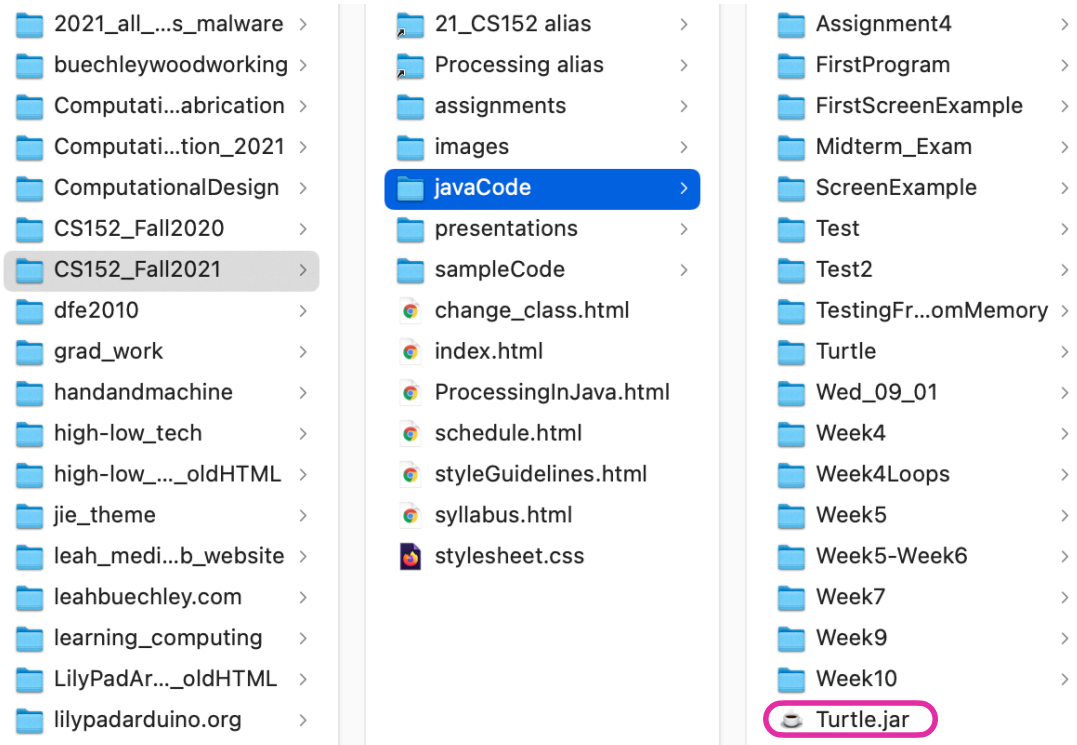
Other Handy Stuff

clearTurtleHistory ()



choose “Keep” at this prompt

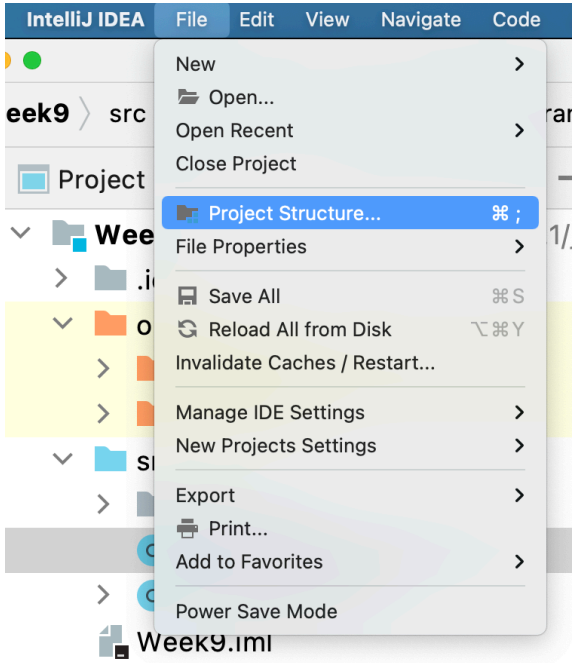
SAVE OR MOVE Turtle.jar INTO YOUR CS152 CODE FOLDER



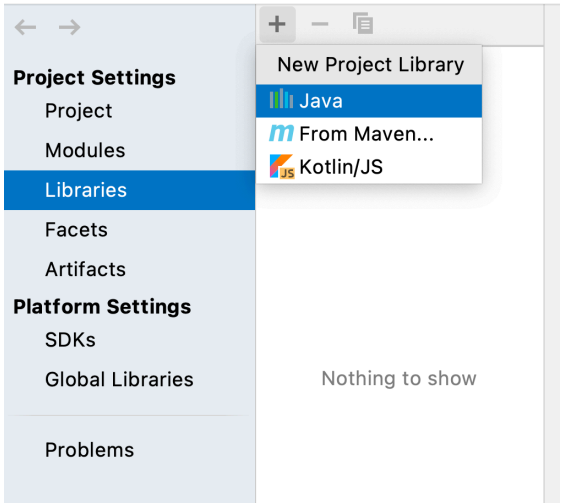
questions?

IN IntelliJ

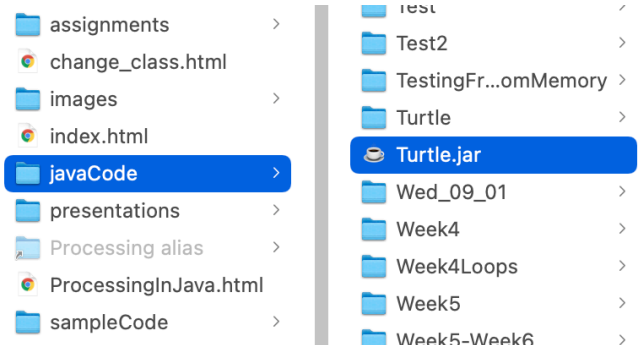
IN IntelliJ



open Project Structure

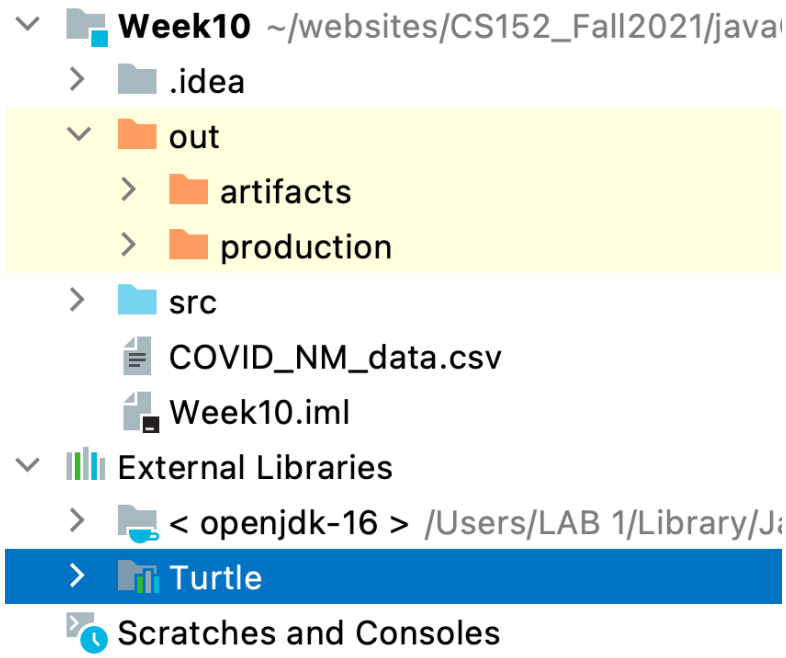


Choose Libraries & click +
Select “Java”



Browse to Turtle.jar and click “OK”
Click “OK” again

IN IntelliJ



Turtle should now appear under External Libraries

questions?

USING THE LIBRARY IN A PROGRAM

IN BasicPanel.java

```
public class BasicPanelPrep extends JPanel implements KeyListener, MouseMotionListener, MouseListener {
    int width;  //window width
    int height; //window height
    char keyPressed;
    int keyCode;
    int mouseX, mouseY;
    Turtle t;

    BasicPanelPrep(int width, int height) {
        this.width = width;
        this.height = height;
        Dimension d = new Dimension(width, height);
        setPreferredSize(d);
        //add listeners for keyboard and mouse interaction
        addKeyListener(this);
        addMouseMotionListener(this);
        addMouseListener(this);
        setFocusable(true);
        requestFocusInWindow();
        setVisible(true);
        t = new Turtle(this);
    }
```

add a Turtle instance variable

initialize the variable
create the Turtle object

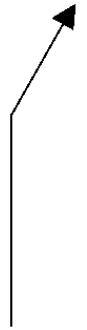
IN paintComponent

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    //turtle drawing
    t.clearTurtleHistory();
    t.forward(100);
    t.right(30);
    t.forward(50);
    t.drawPath(g);
    t.drawTurtle(g);
}
```

clear previous turtle path

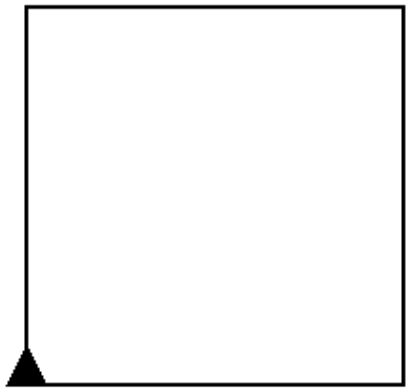
after you're done with commands
draw the Turtle path

draw the Turtle (optional)



A MORE COMPLEX TURTLE PROGRAM

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    t.clearTurtleHistory();
    for (int i=0;i<4;i++) {
        t.forward(100);
        t.right(90);
    }
    t.drawPath(g);
    t.drawTurtle(g);
}
```



questions?

BACK TO TREES

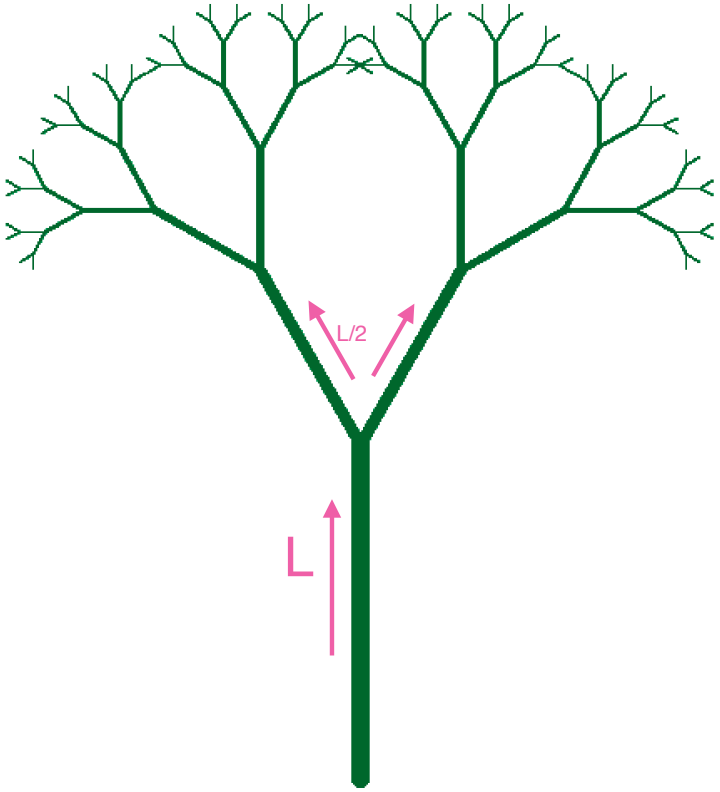
A TREE PROGRAM in PSEUDOCODE

```
tree (L, A)
  move forward L

  rotate left an angle A
  move forward L/2
  move backward L/2
  rotate right angle A

  rotate right an angle A
  move forward L/2
  move backward L/2
  rotate left an angle A

  move backward length L
```



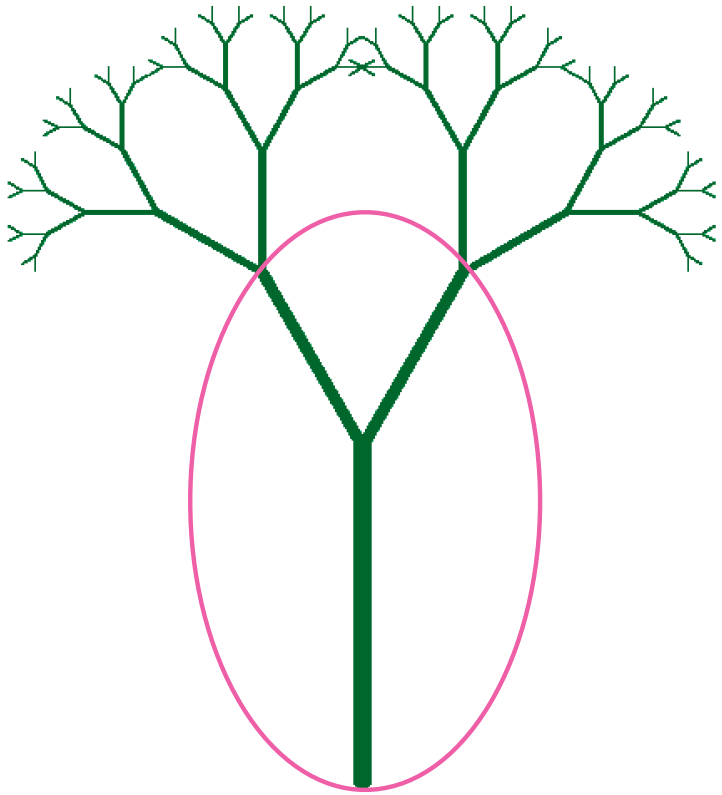
A TREE PROGRAM in PSEUDOCODE

```
tree (L, A)
  move forward L

  rotate left an angle A
  move forward L/2
  move backward L/2
  rotate right angle A

  rotate right an angle A
  move forward L/2
  move backward L/2
  rotate left an angle A

  move backward length L
```



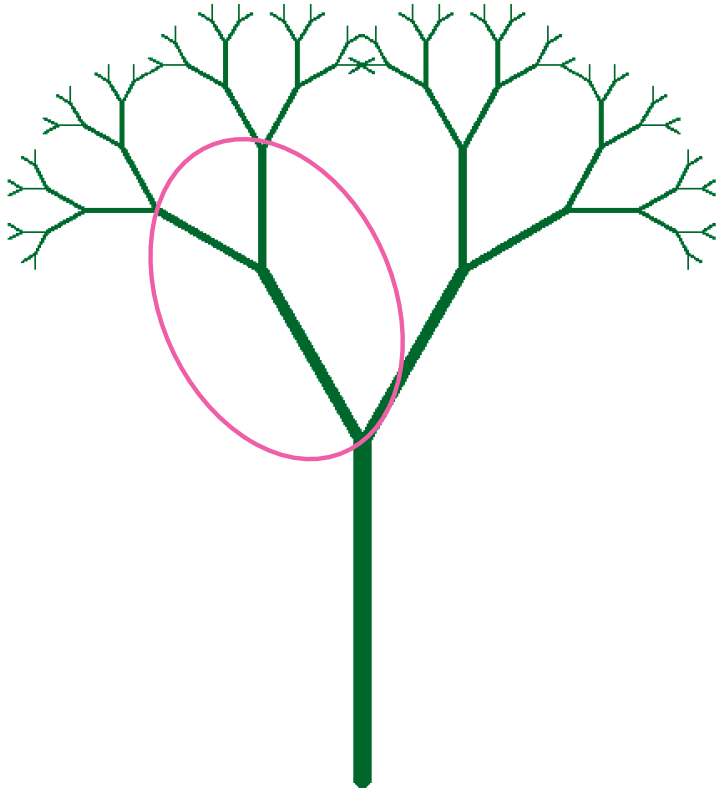
A TREE PROGRAM in PSEUDOCODE

```
tree (L, A)
  move forward L

  rotate left an angle A
  move forward L/2
  move backward L/2
  rotate right angle A

  rotate right an angle A
  move forward L/2
  move backward L/2
  rotate left an angle A

  move backward length L
```



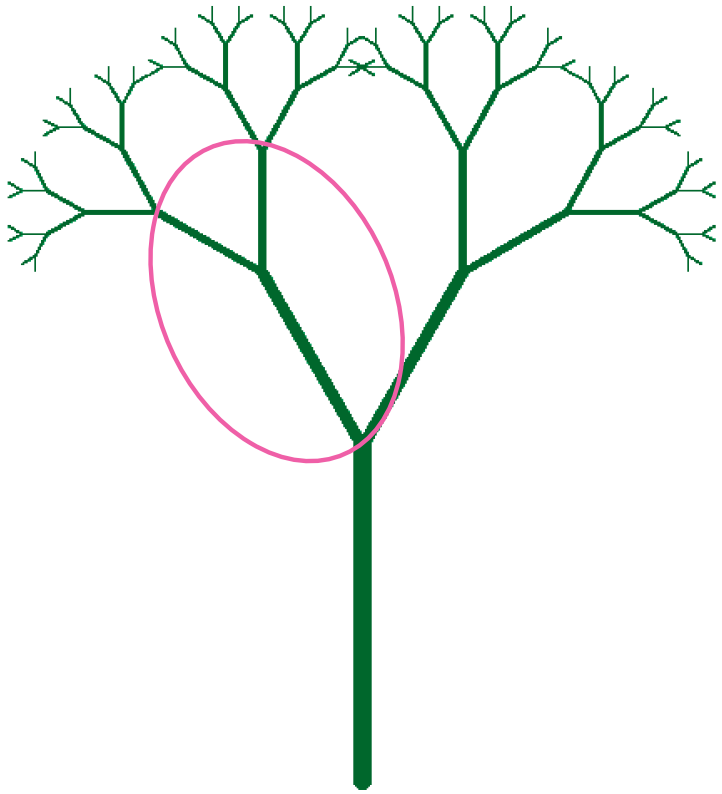
A TREE PROGRAM in PSEUDOCODE

```
tree (L, A)
  move forward L

  rotate left an angle A
  tree (L/2, A)
  rotate right angle A

  rotate right an angle A
  tree (L/2, A)
  rotate left an angle A

  move backward length L
```



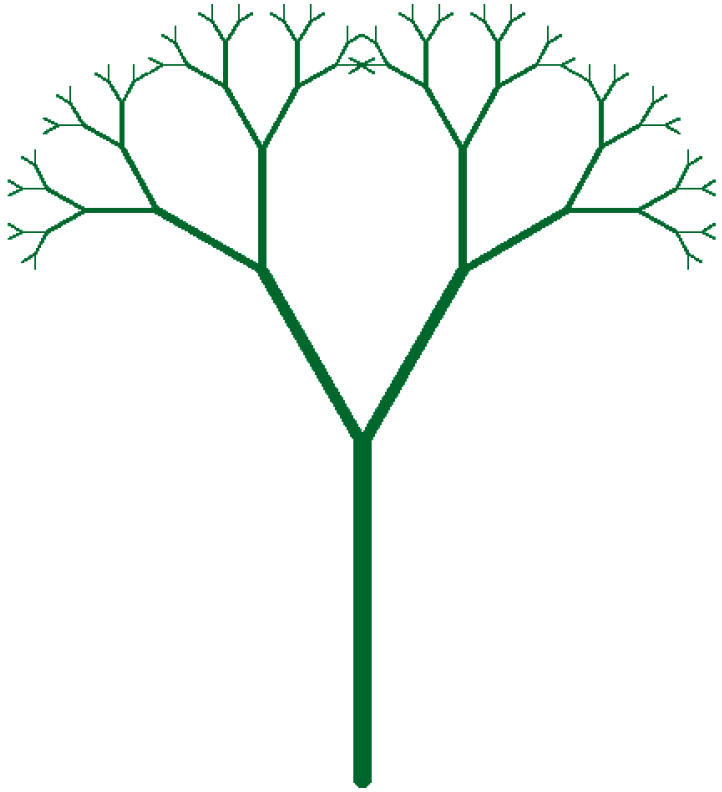
A TREE PROGRAM in PSEUDOCODE
a stopping condition?

```
tree (L, A)
  move forward L

  rotate left an angle A
  tree (L/2, A)
  rotate right angle A

  rotate right an angle A
  tree (L/2, A)
  rotate left an angle A

  move backward length L
```



A TREE PROGRAM in PSEUDOCODE
a stopping condition?

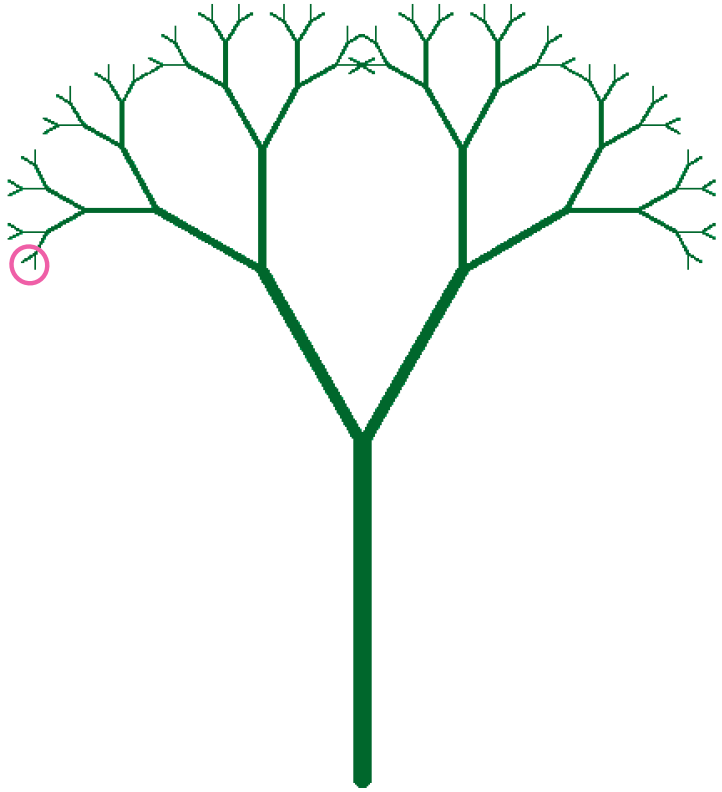
```
tree (L, A)
  if (L < 20)
    STOP

  else
    move forward L

    rotate left an angle A
    tree (L/2, A)
    rotate right angle A

    rotate right an angle A
    tree (L/2, A)
    rotate left an angle A

    move backward length L
```



questions?

A TREE PROGRAM in CODE

```
tree (L, A)
  if (L < 20)
    STOP

  else
    move forward L

    rotate left an angle A
    tree (L/2, A)
    rotate right angle A

    rotate right an angle A
    tree (L/2, A)
    rotate left an angle A

    move backward length L
```

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

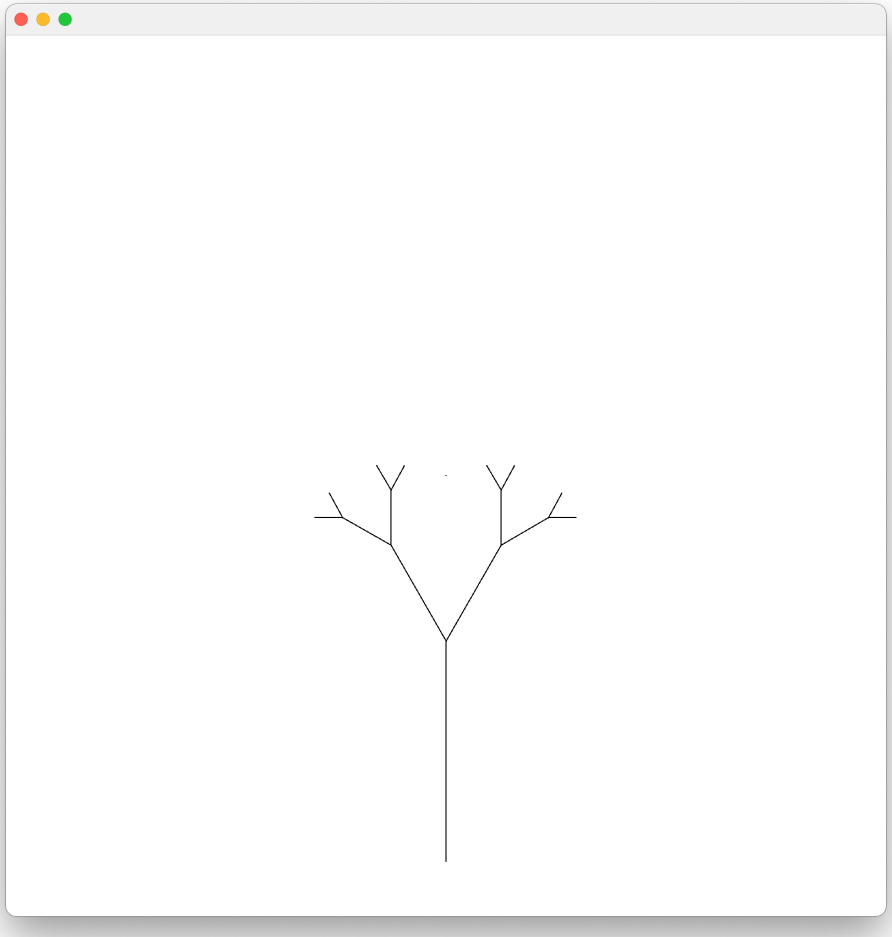
questions?

DRAWING OUR TREE

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    g.setColor(Color.DARK_GRAY);
    t.clearTurtleHistory();
    drawTree(200,30);
    t.drawPath(g);
}
```

call tree method with a size (200) and an angle (30)

DRAWING OUR TREE



ANIMATING OUR TREE DRAWING

PaintComponent

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    g.setColor(Color.DARK_GRAY);
    t.clearTurtleHistory();
    t.penUp();
    t.backward(height/2-10);
    t.penDown();
    t.setStroke(5);
    drawTree(200,30);
    t.drawPath(g);
}
```

PaintComponent

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    g.setColor(Color.DARK_GRAY);
    t.drawStep(g);
}
```

each time you paint, draw
one step of the turtle drawing

ANIMATING OUR TREE DRAWING

```
public static void main(String[] args) {
    BasicPanel panel = new BasicPanel(800,800);
    MyFrame f = new MyFrame(panel);
    panel.tree(200,30);
}
```

call tree method with a size (200) and an angle (30)

REPAINT IS CALLED IN keyTyped
THIS WILL ANIMATE

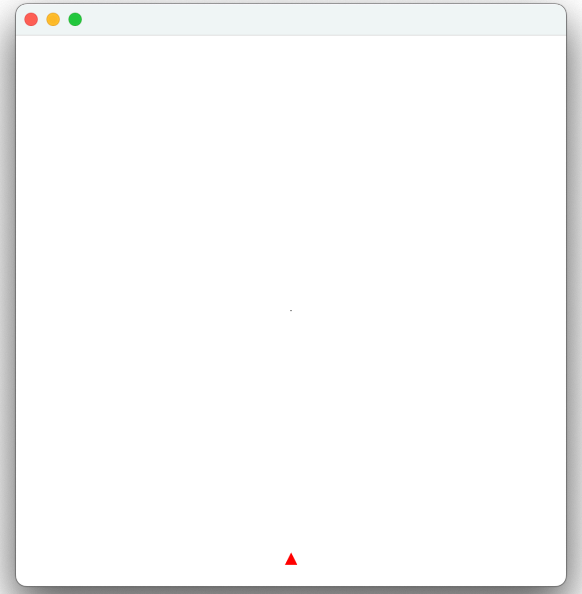
```
public void keyTyped(KeyEvent e) {
    keyPressed = e.getKeyChar();
    System.out.println(keyPressed);
    //if spacebar is pressed, repaint
    if (keyPressed==' ')
        repaint();
}
```

questions?

LET'S STEP THROUGH THE PROGRAM AS IT ANIMATES

ANIMATING OUR TREE DRAWING

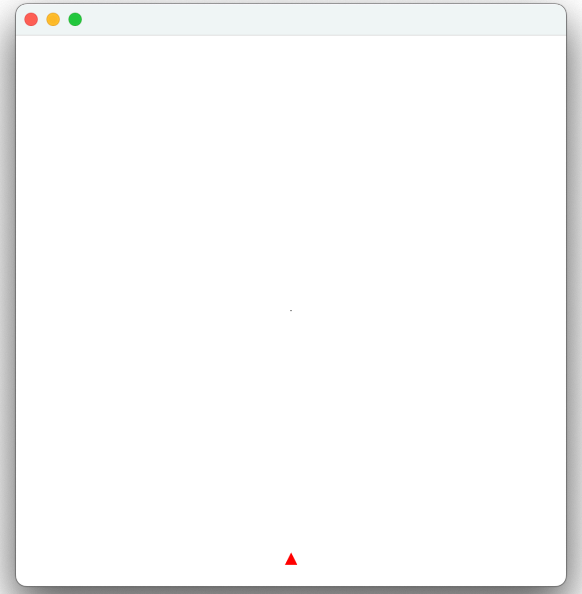
```
public static void main(String[] args) {
    BasicPanel panel = new BasicPanel(800,800);
    MyFrame f = new MyFrame(panel);
    panel.moveToCenterBottom();
    panel.drawTree(200,30);
}
```



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

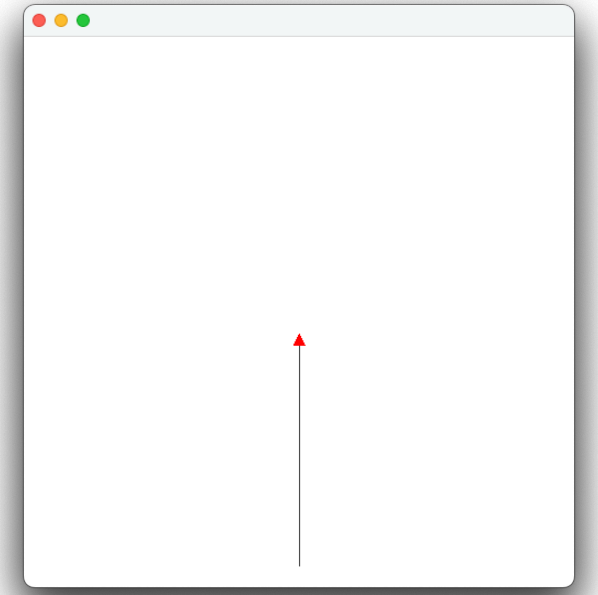
drawTree (200, 30) ← stack



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

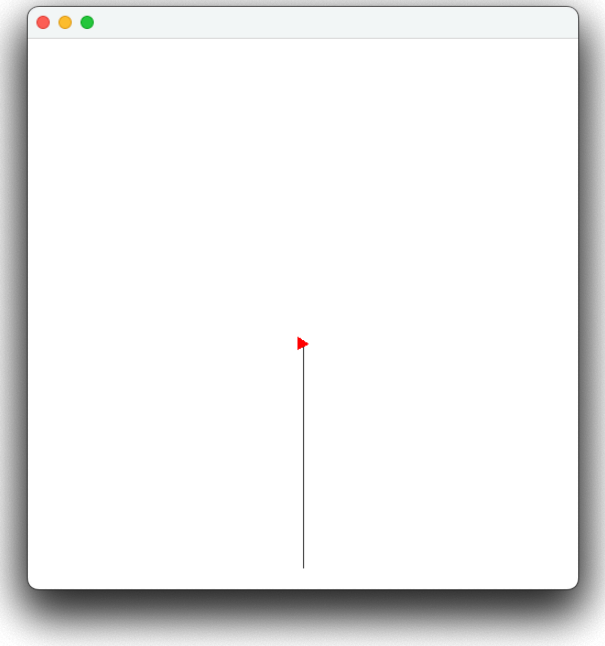
drawTree (200, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

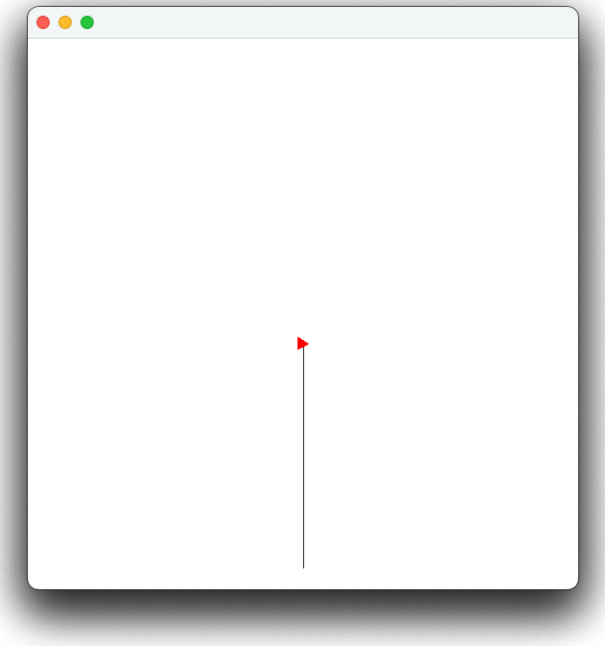
drawTree (200, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

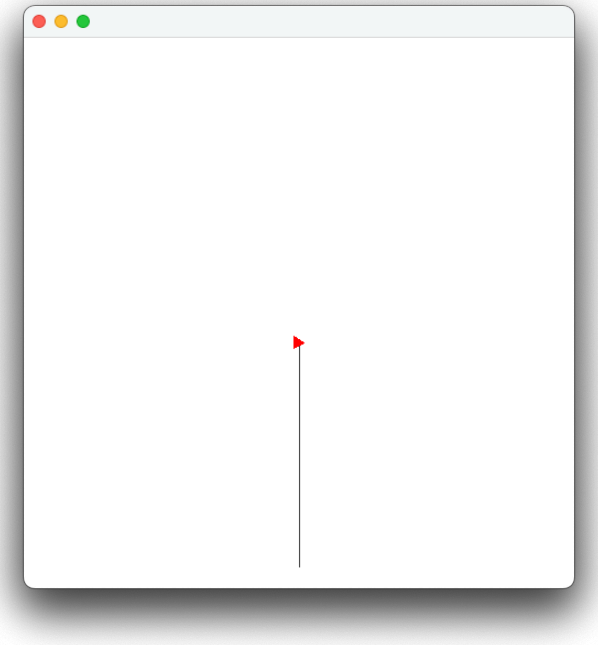
drawTree (200, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

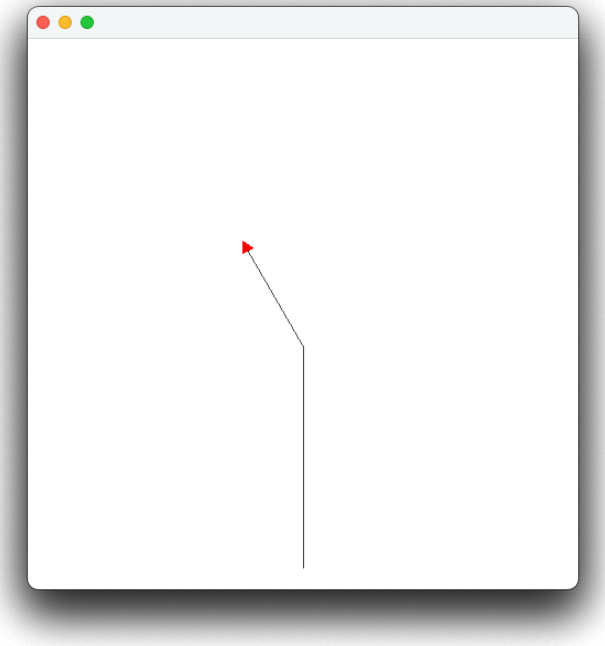
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

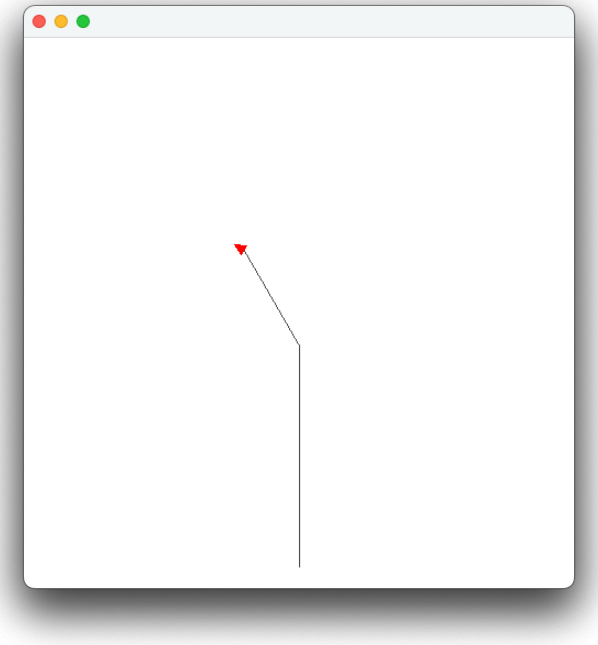
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

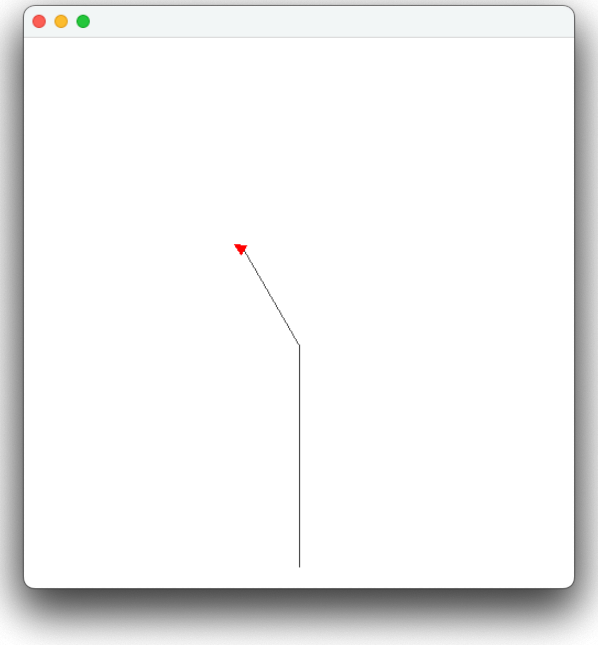
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

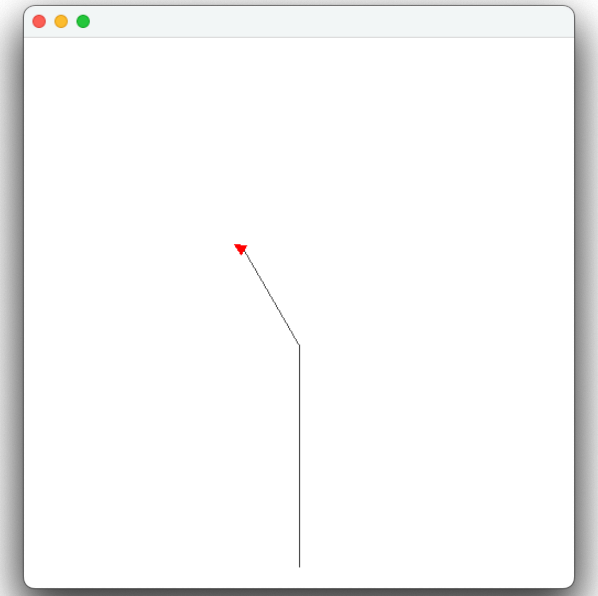
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

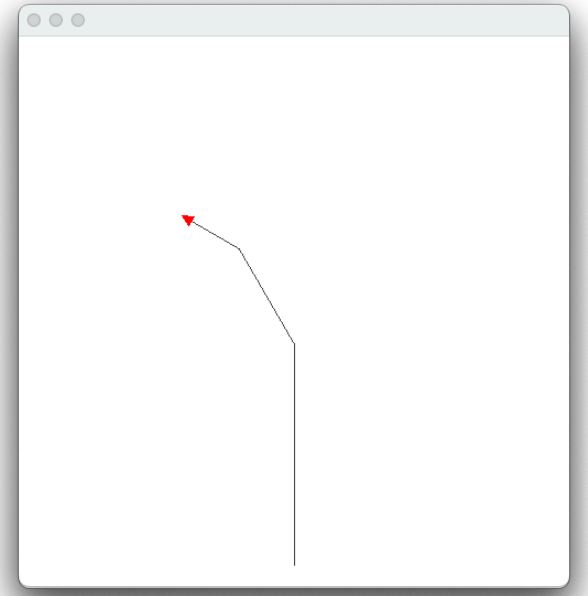
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

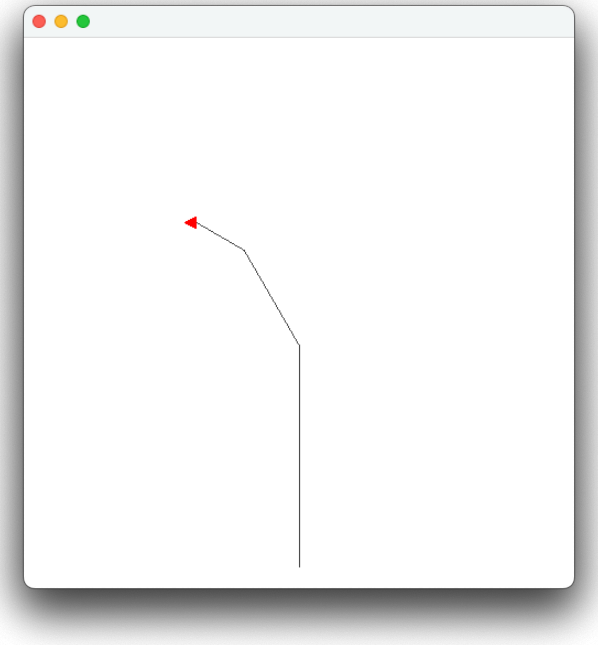
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

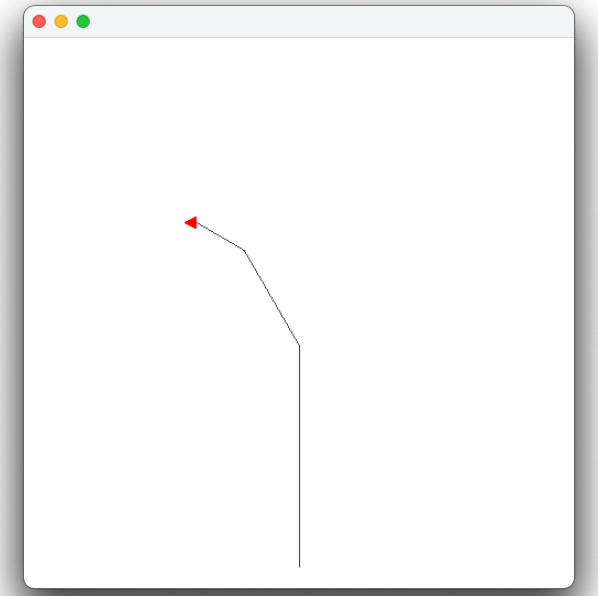
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

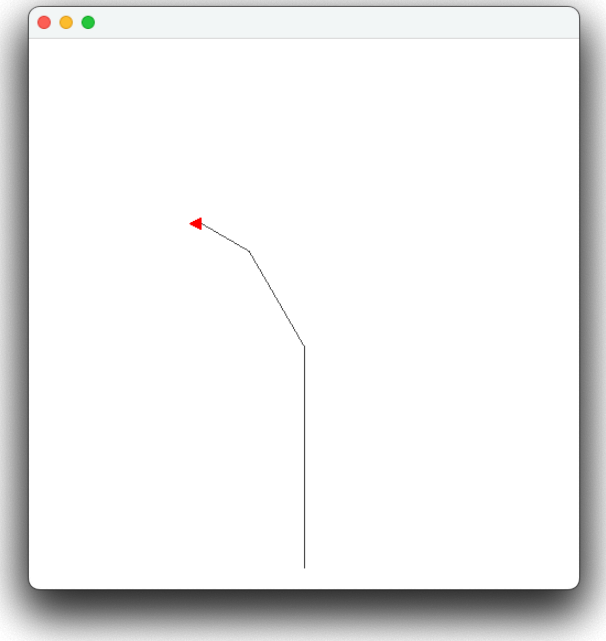
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

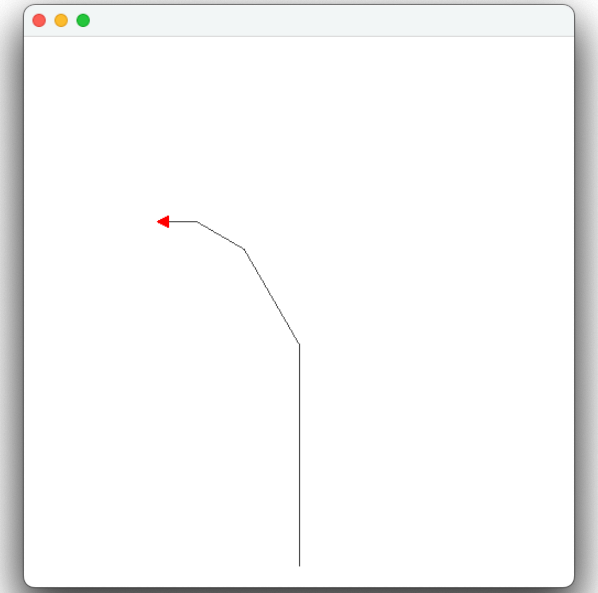
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

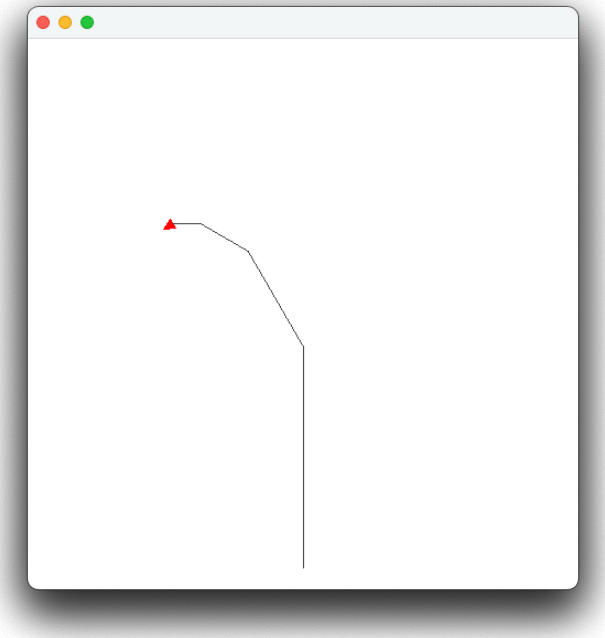
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

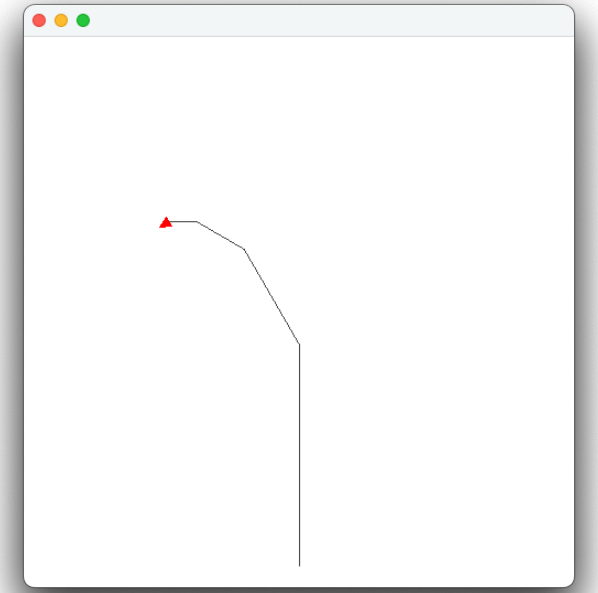
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

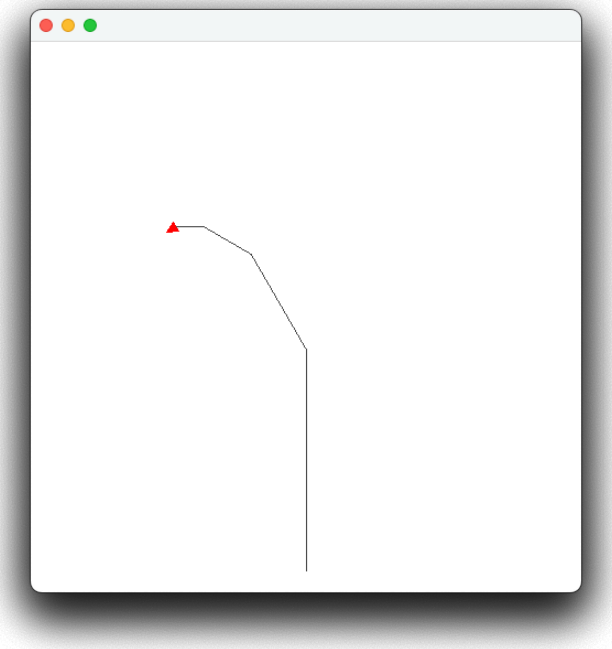
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

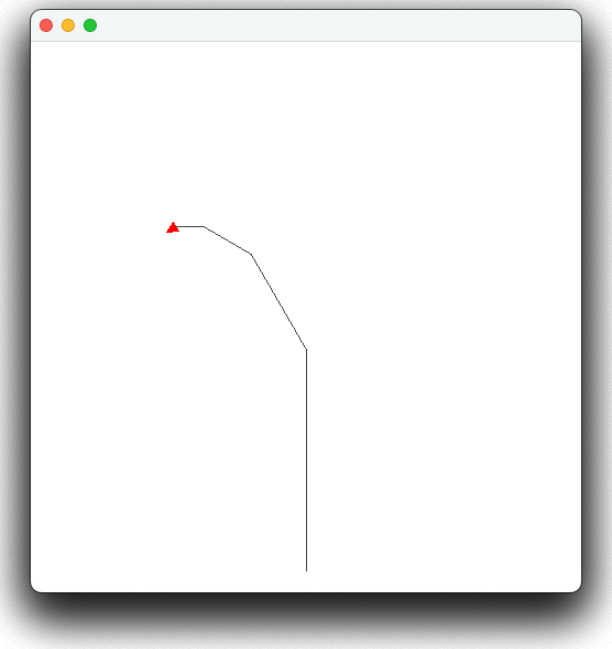
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

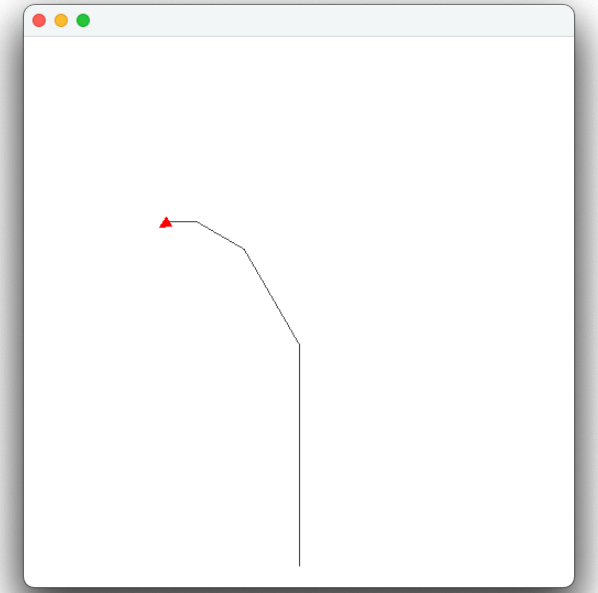
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

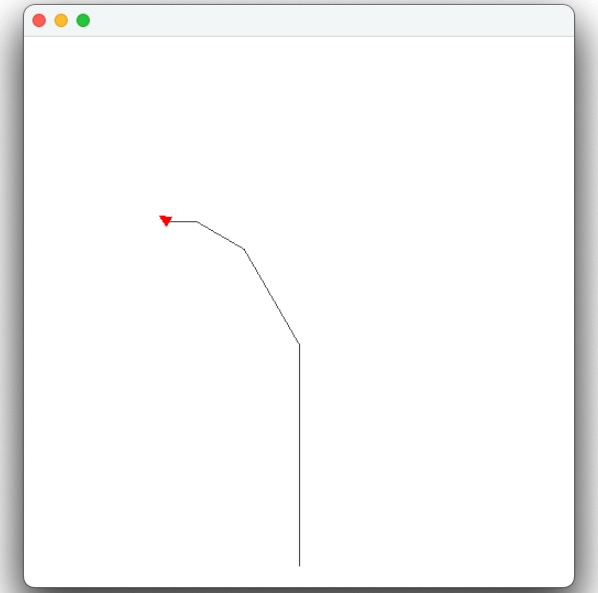
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

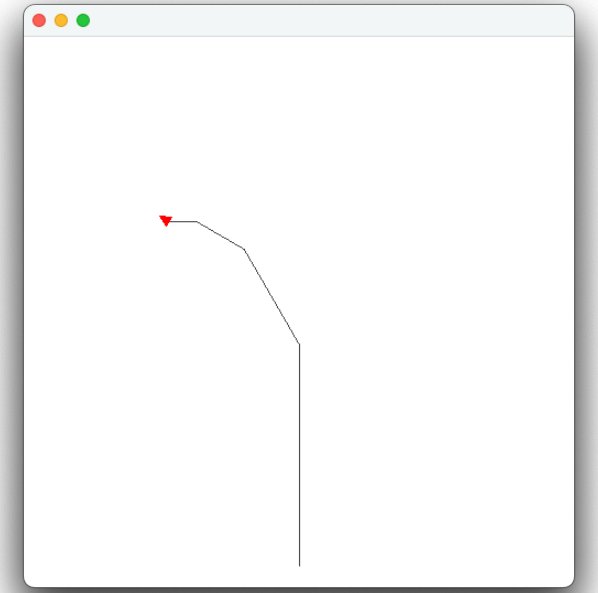
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

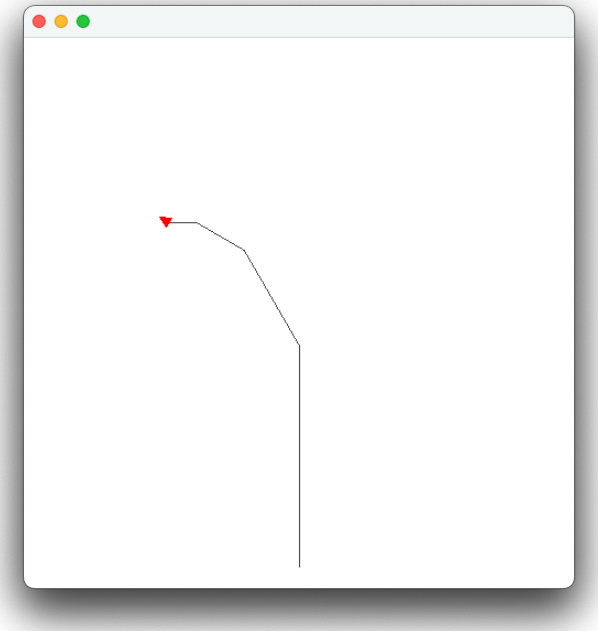
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

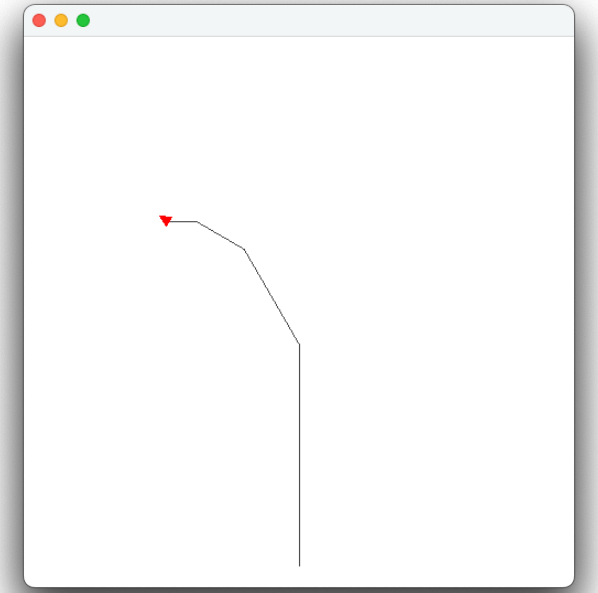
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

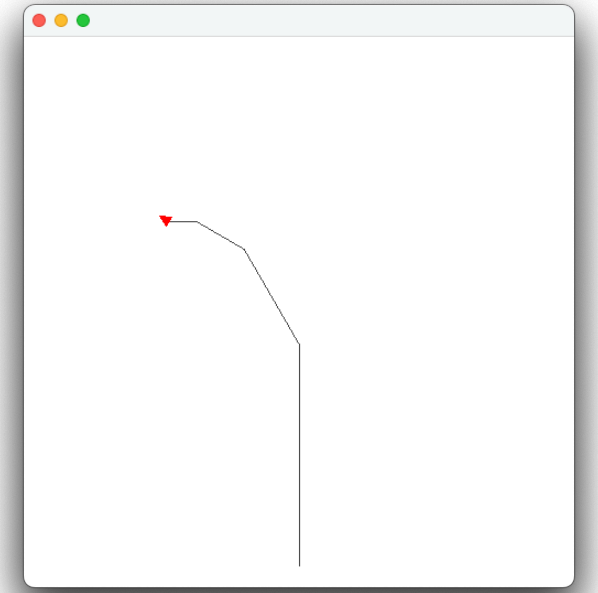
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

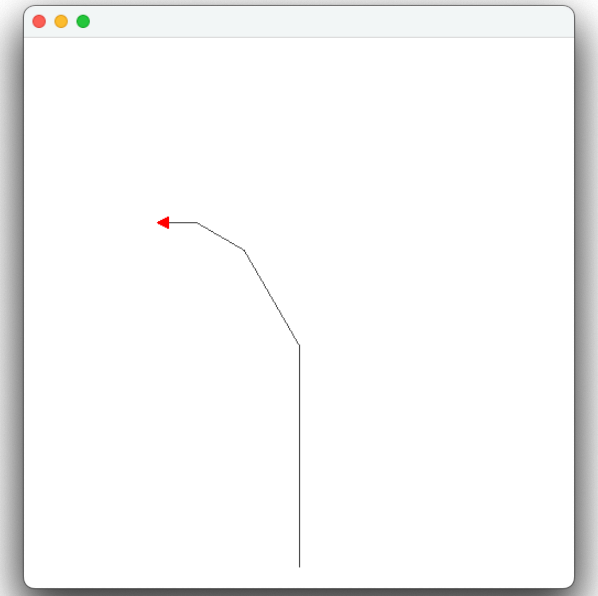
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

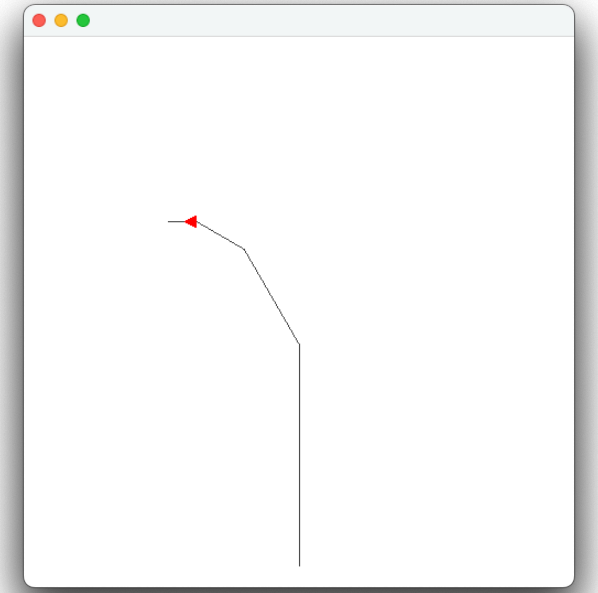
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

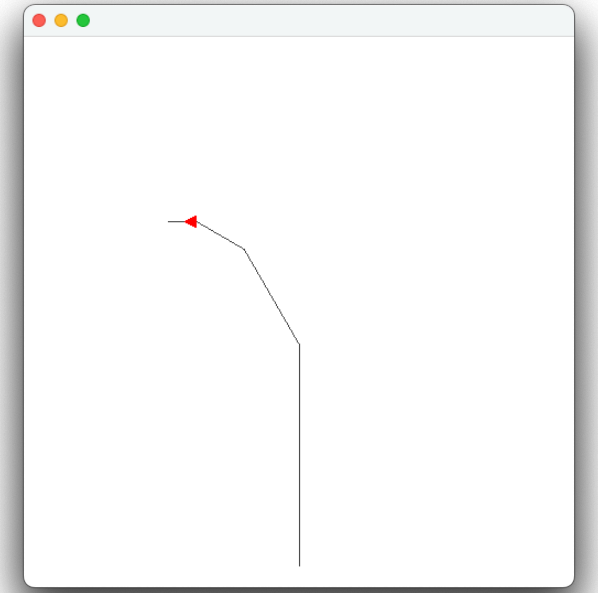
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

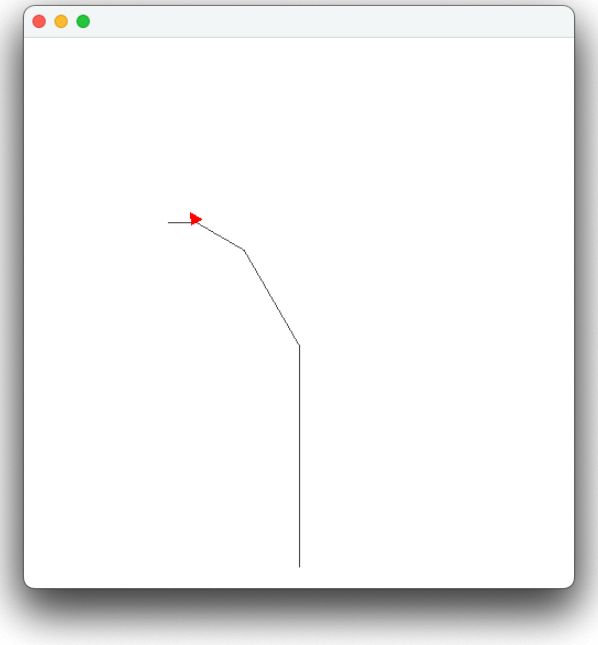
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

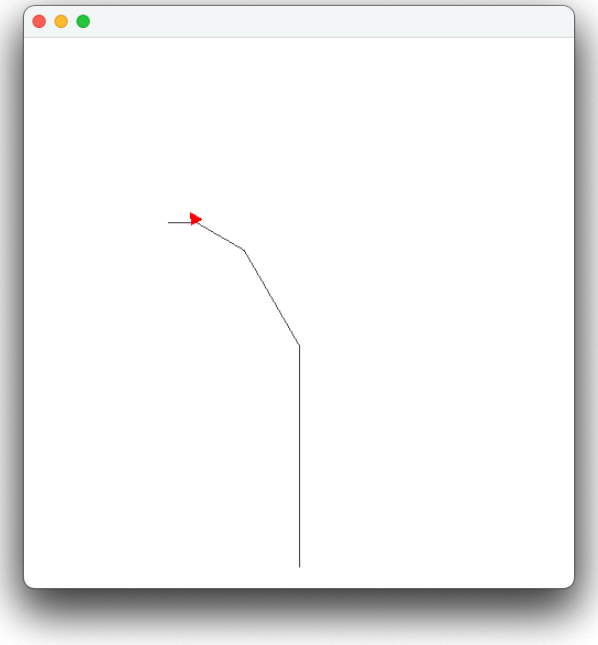
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

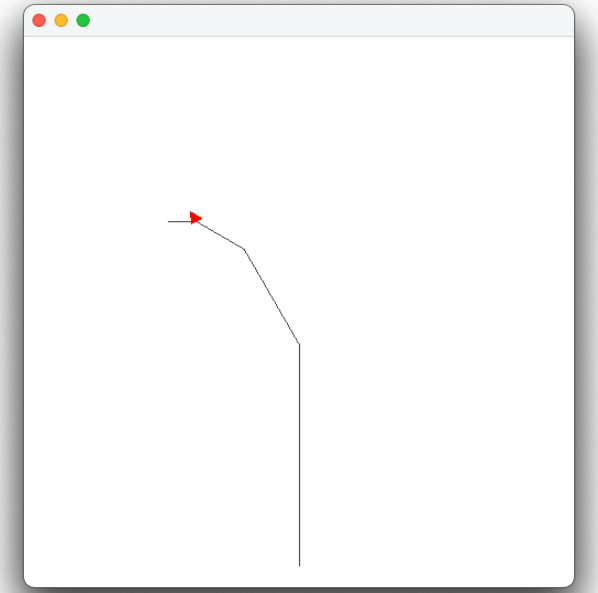
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

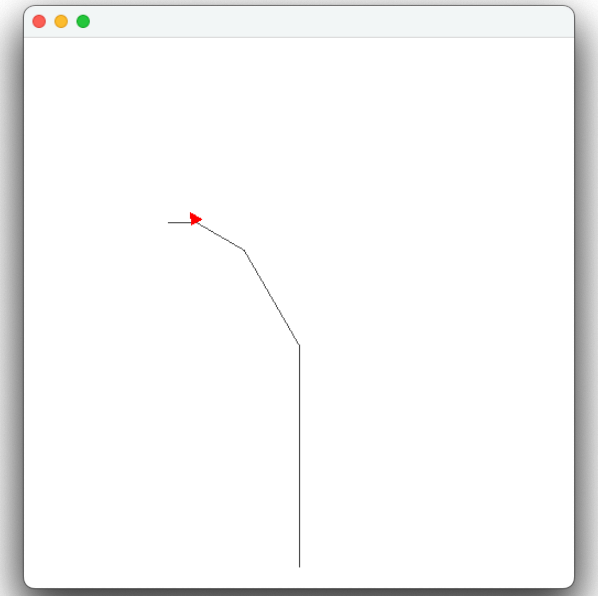
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

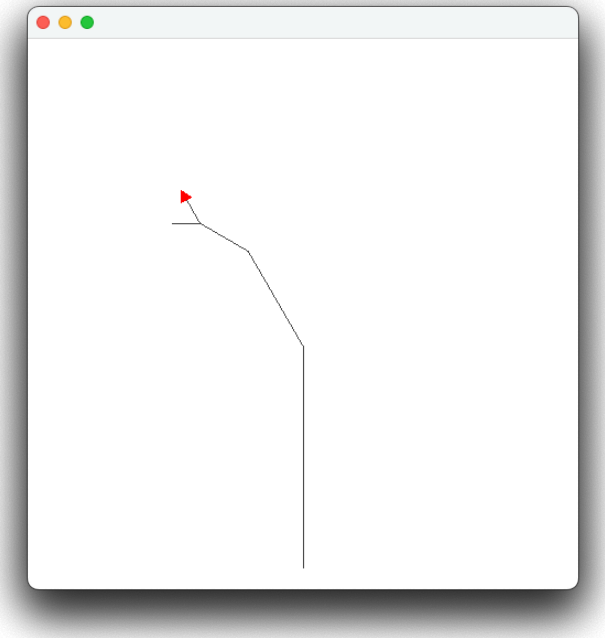
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

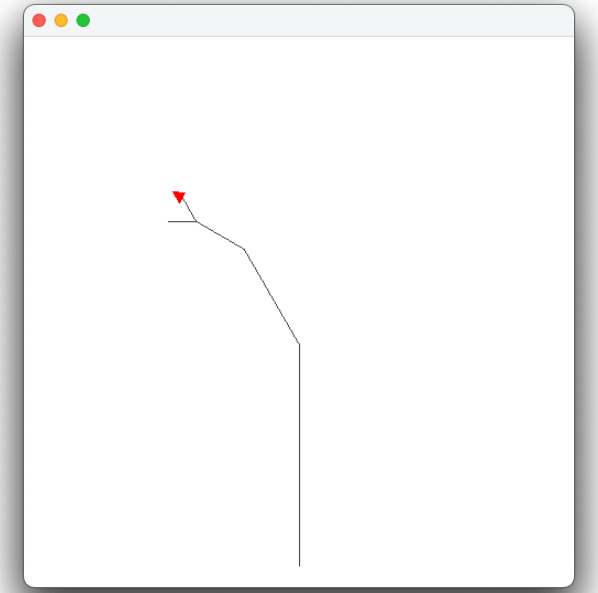
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

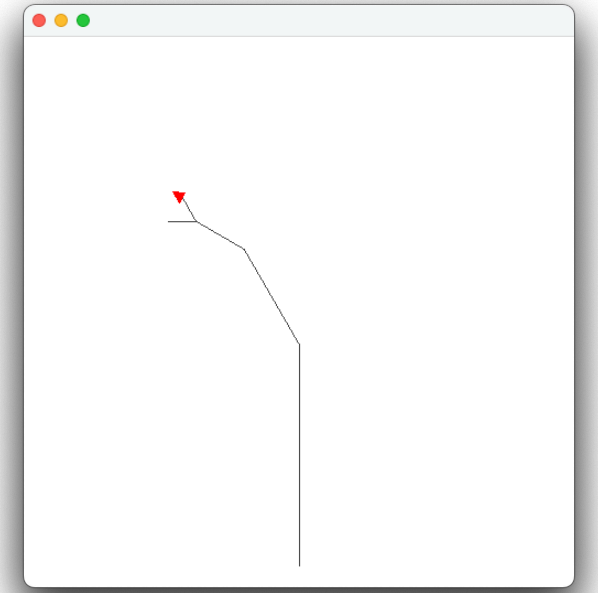
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

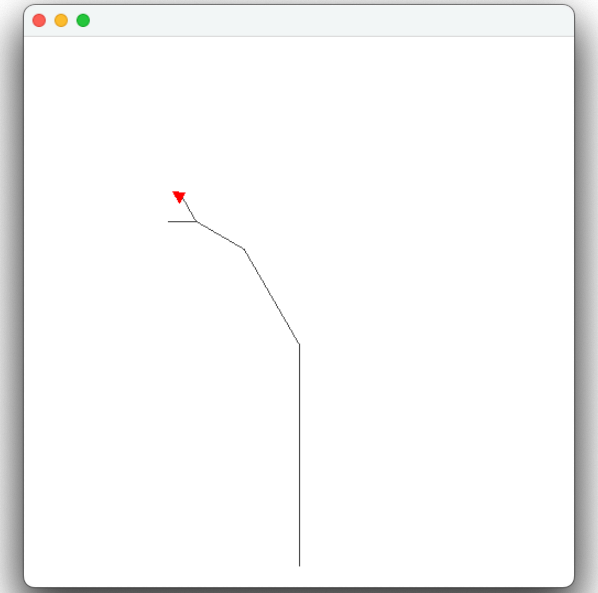
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

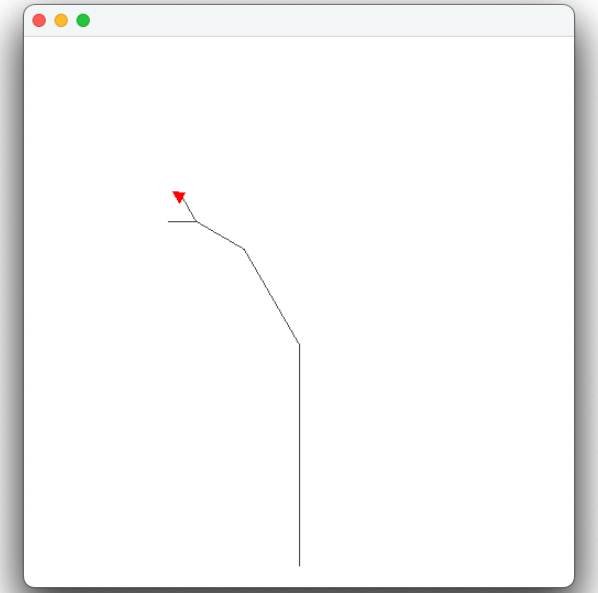
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

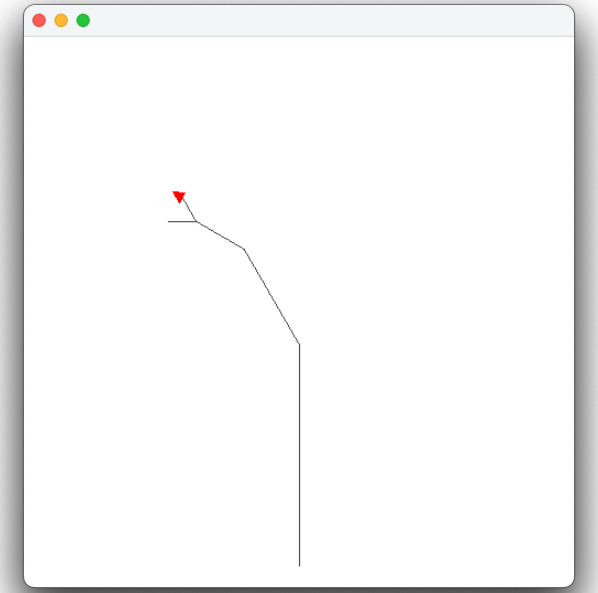
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

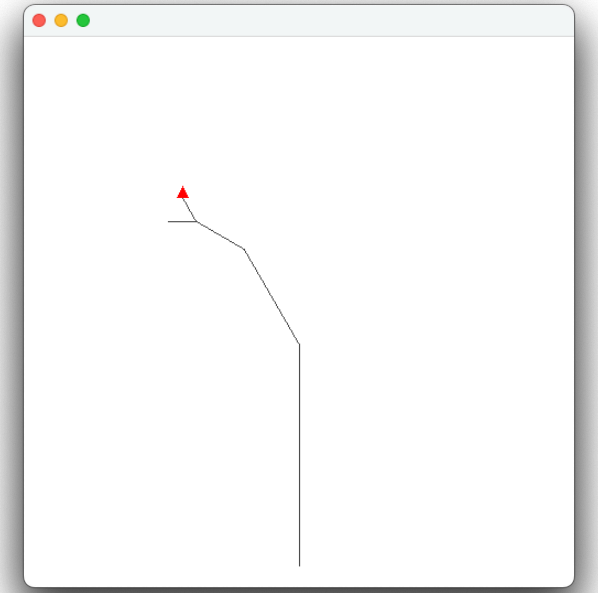
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

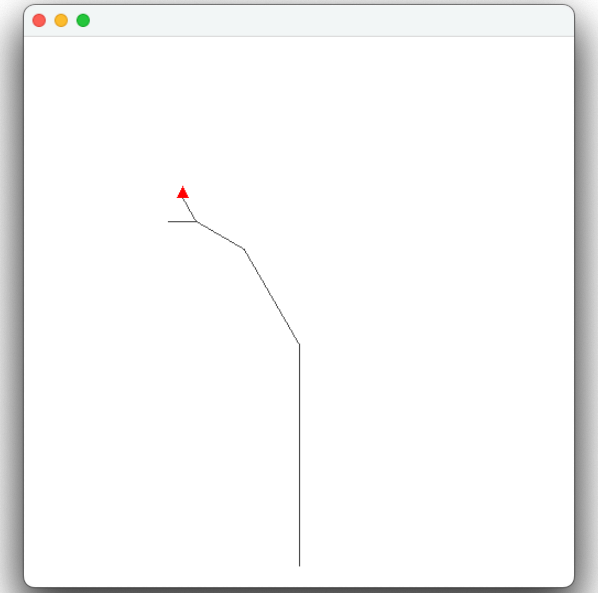
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

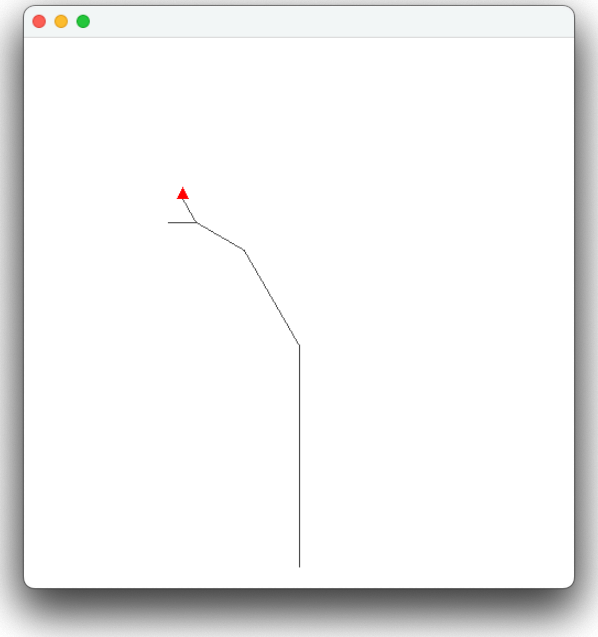
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

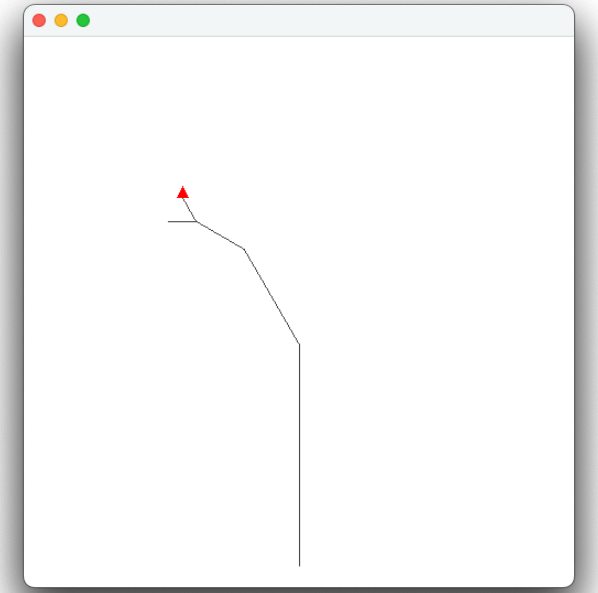
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

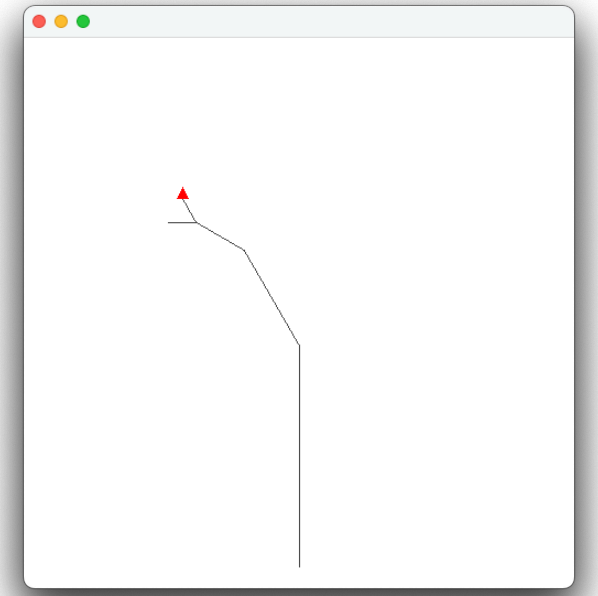
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)
drawTree (12, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

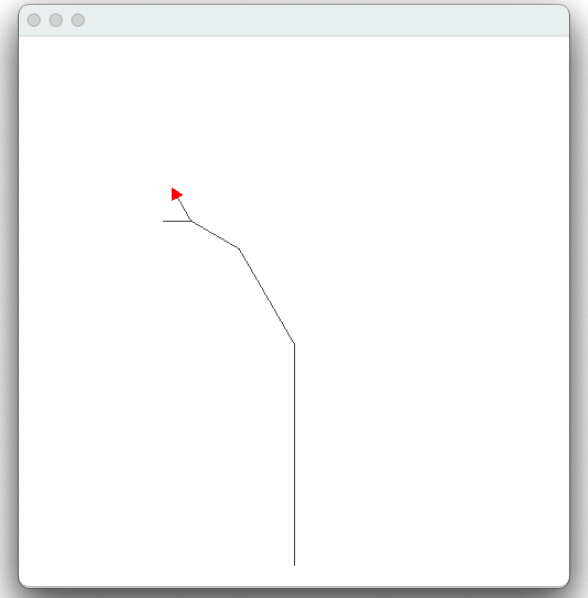
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

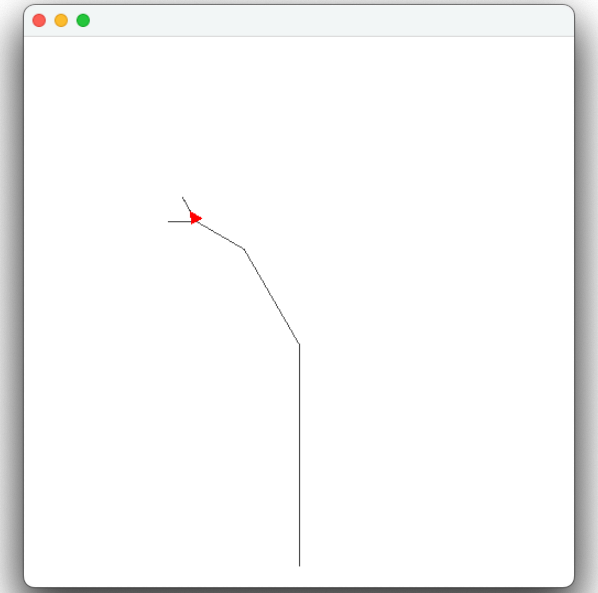
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

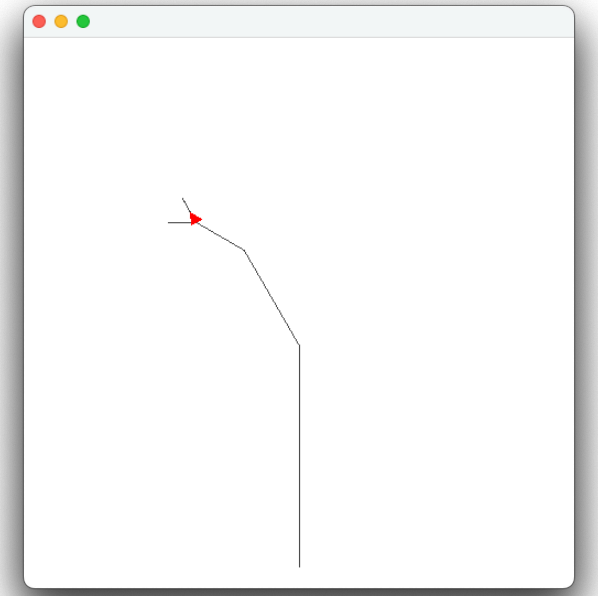
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)
drawTree (25, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

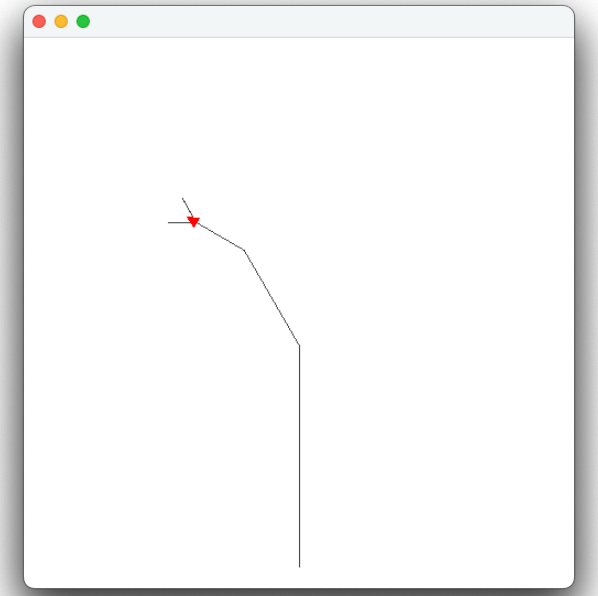
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

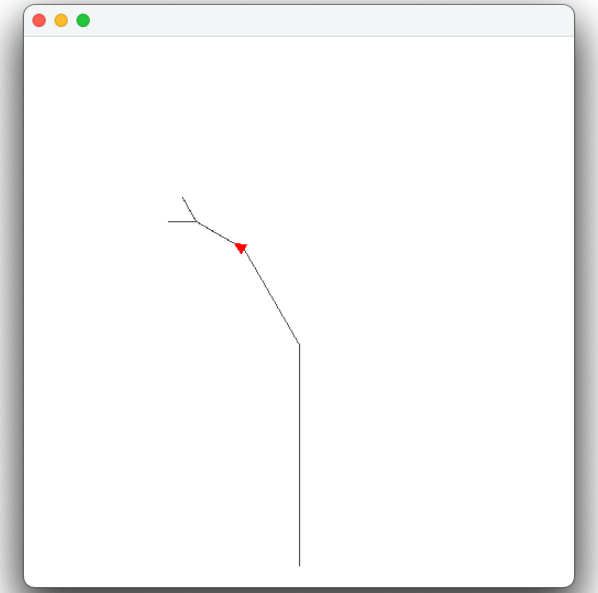
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

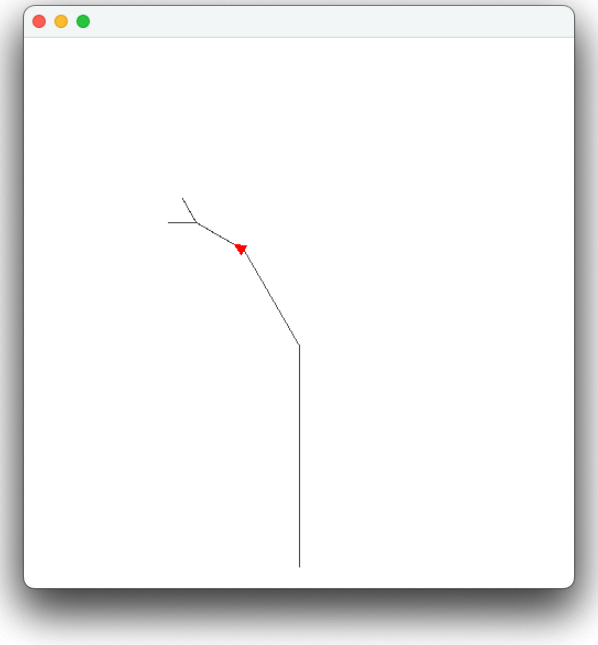
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

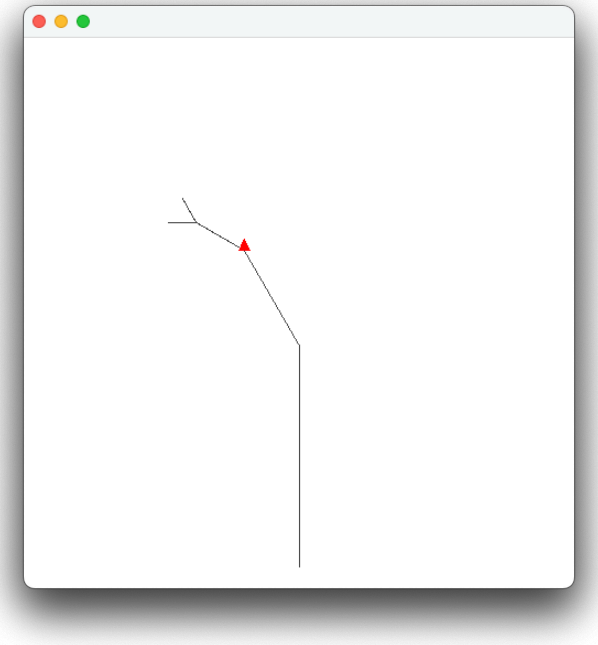
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

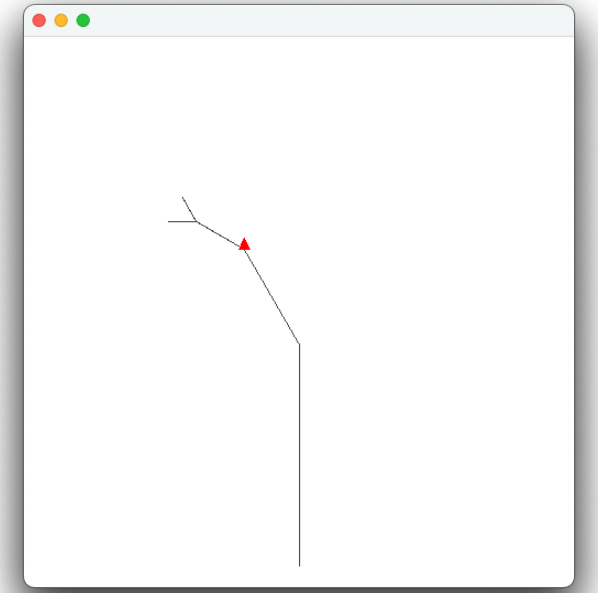
drawTree (200, 30)
drawTree (100, 30)



ANIMATING OUR TREE DRAWING

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

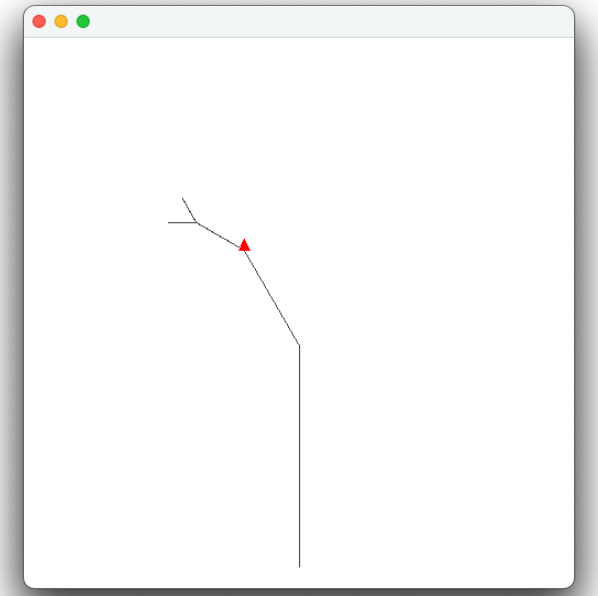
drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ANIMATING OUR TREE DRAWING

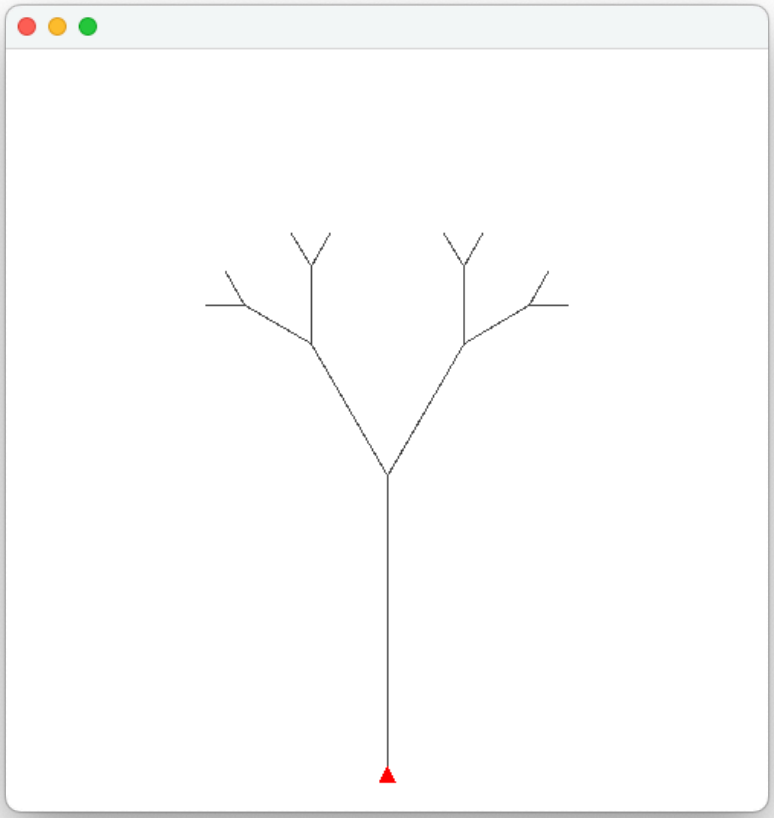
```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

drawTree (200, 30)
drawTree (100, 30)
drawTree (50, 30)



ETC…

ANIMATING OUR TREE DRAWING



LET'S US SEE THE ALGORITHM IN ACTION!

CAN ALSO RUN IN DEBUGGER, BUT LOSE VISUALIZATION

questions?

PLAYING WITH PARAMETERS

1st: DRAW INSTEAD OF ANIMATE

PaintComponent

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    g.setColor(Color.DARK_GRAY);
    t.drawStep(g);
}
```

each time you paint, draw
one step of the turtle drawing

PaintComponent

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);
    g.setColor(Color.DARK_GRAY);
    t.drawPath(g);
}
```

each time you paint, draw the entire turtle drawing

questions?

PLAYING WITH PARAMETERS

OUR TREE METHOD

```
void drawTree (double length, double angle) {
    if (length<20)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/2, angle); //left branch
        t.right(angle*2);
        drawTree(length/2, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

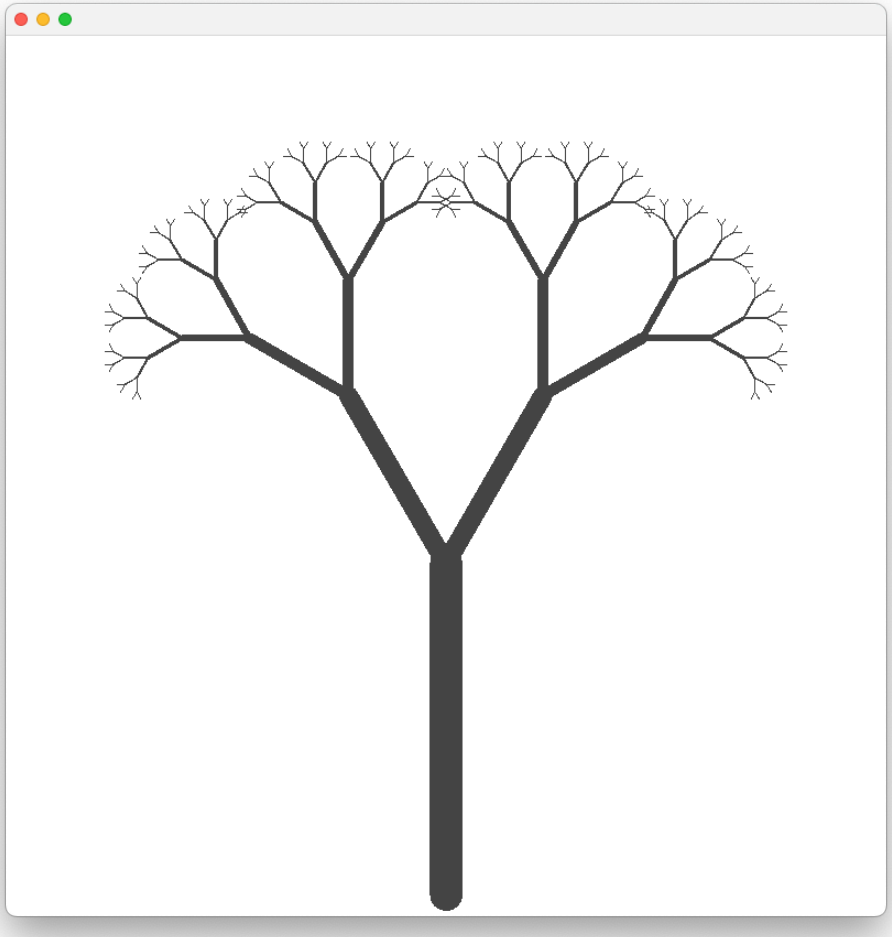
OUR TREE METHOD

```
void drawTree (double length, double angle) {
    if (length<5)
        return;
    else {
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/1.7, angle); //left branch
        t.right(angle*2);
        drawTree(length/1.7, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

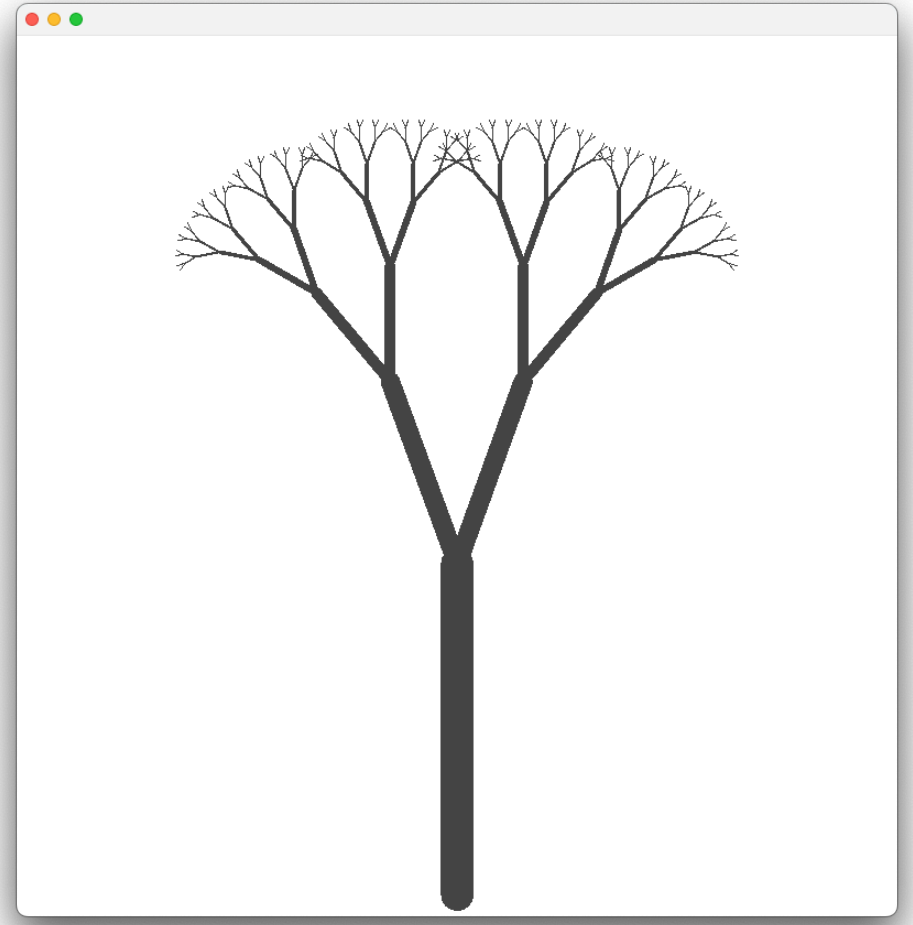
ADD STROKE

```
void drawTree (double length, double angle) {
    if (length<5)
        return;
    else {
        t.setStroke(length/10);
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/1.7, angle); //left branch
        t.right(angle*2);
        drawTree(length/1.7, angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

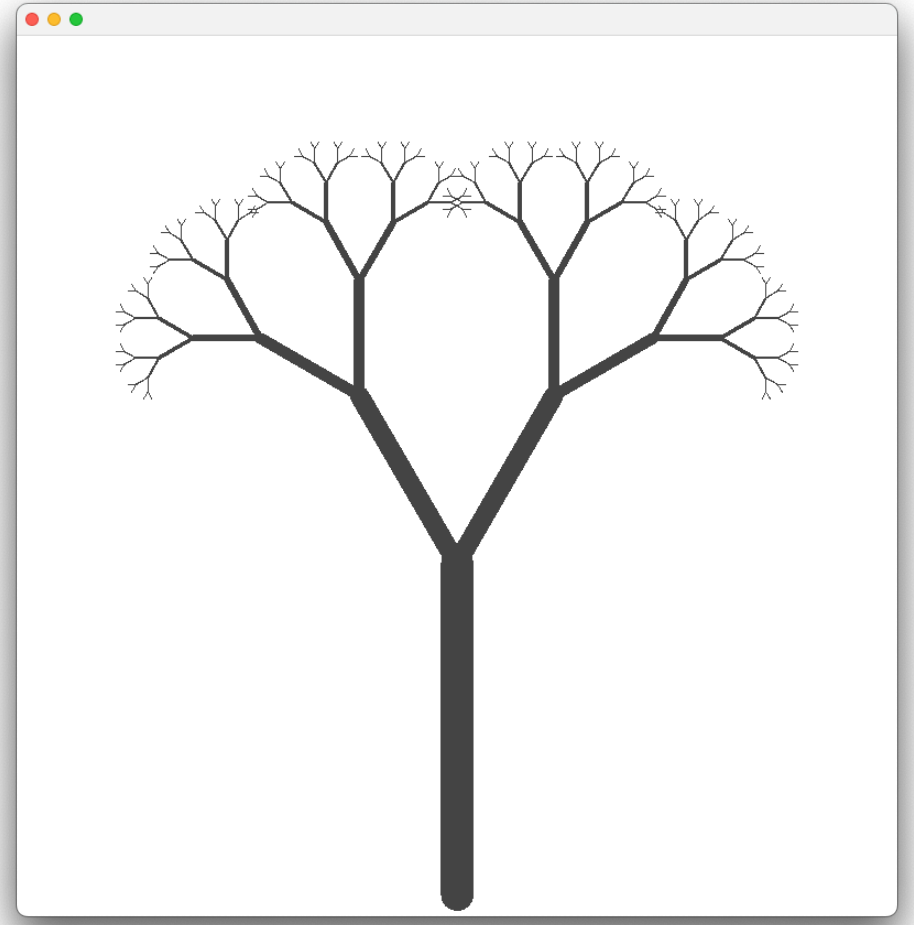
```
panel.drawTree(300,30);
```



```
panel.drawTree(300,20);
```



```
panel.drawTree(300,50);
```



questions?

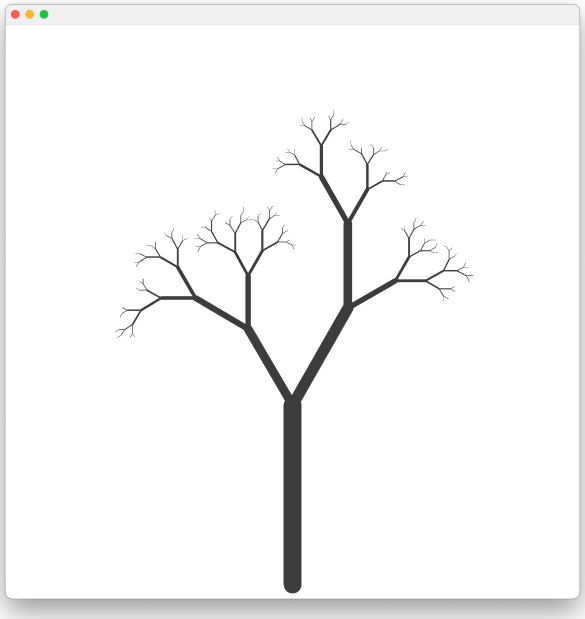
ADDING SOME RANDOMNESS

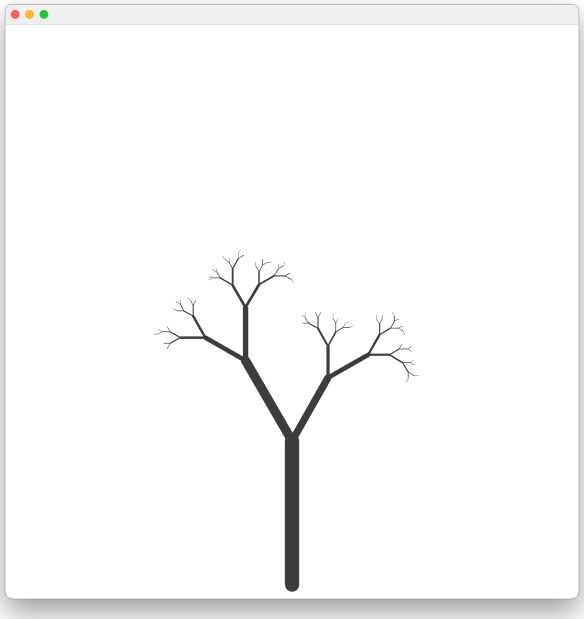
OUR TREE METHOD

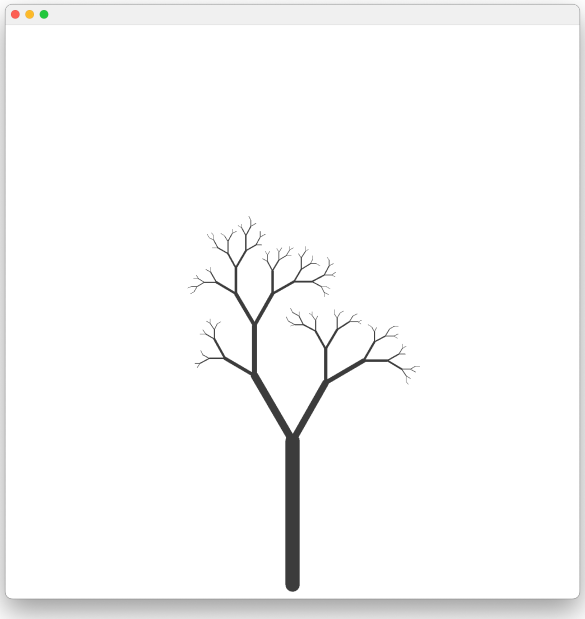
```
void drawTree (double length, int angle) {
    if (length<5)
        return;
    else {
        t.setStroke(length/10);
        t.forward(length);  //trunk
        t.left(angle);
        drawTree(length/(1.1+Math.random()), angle); //left branch
        t.right(angle*2);
        drawTree(length/(1.1+Math.random()), angle); //right branch
        t.left(angle);
        t.backward(length);
    }
}
```

randomly change size of branches
via scale factor

```
panel.drawTree(200,30);
```







A NEW TREE WITH EACH SPACEBAR PRESS

IN KeyTyped

```
public void keyTyped(KeyEvent e) {
    keyPressed = e.getKeyChar();
    System.out.println(keyPressed);
    //if spacebar is pressed, generate and draw a new tree
    if (keyPressed==' ') {
        t.clearTurtleHistory();
        drawTree(200,30);
        repaint();
    }
}
```

questions?

L-SYSTEMS

L-Systems

developed by Aristid Lindenmayer in 1964

Lindenmayer was a Hungarian botanist and biologist

developed as a way to understand plant growth

“Algorithmic Beauty of Plants” published in 1990

L-Systems

A set of rules that turn a simple starting word into a more complex word by replacing symbols with more complex patterns.

Example:

starting word: F
rule: F-F++F-F

F
F-F++F-F
F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F

L-Systems and Turtle Geometry

Visualize the pattern by translating it into actions carried out by the turtle. Each symbol is translated into an action

Example:

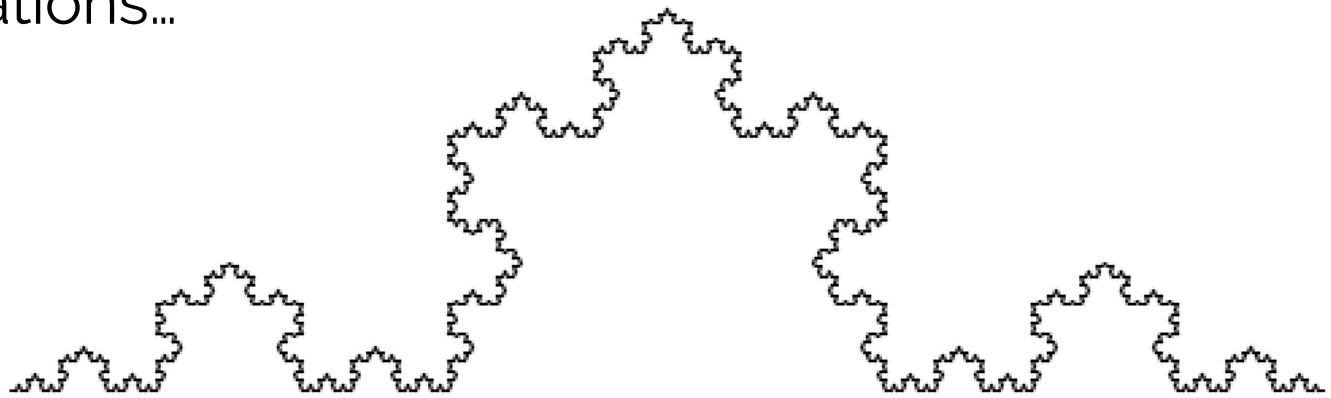
F = move forward an amount (100)
- = turn left an angle (60°)
+ = turn right an angle (60°)

F
F-F++F-F
F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F

L-Systems and Turtle Geometry

F = move forward an amount (100)
- = turn left an angle (60°)
+ = turn right an angle (60°)

more iterations...



questions?

MORE ON MONDAY

QUIZ 3
DEADLINE SUNDAY

Thank you!

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/