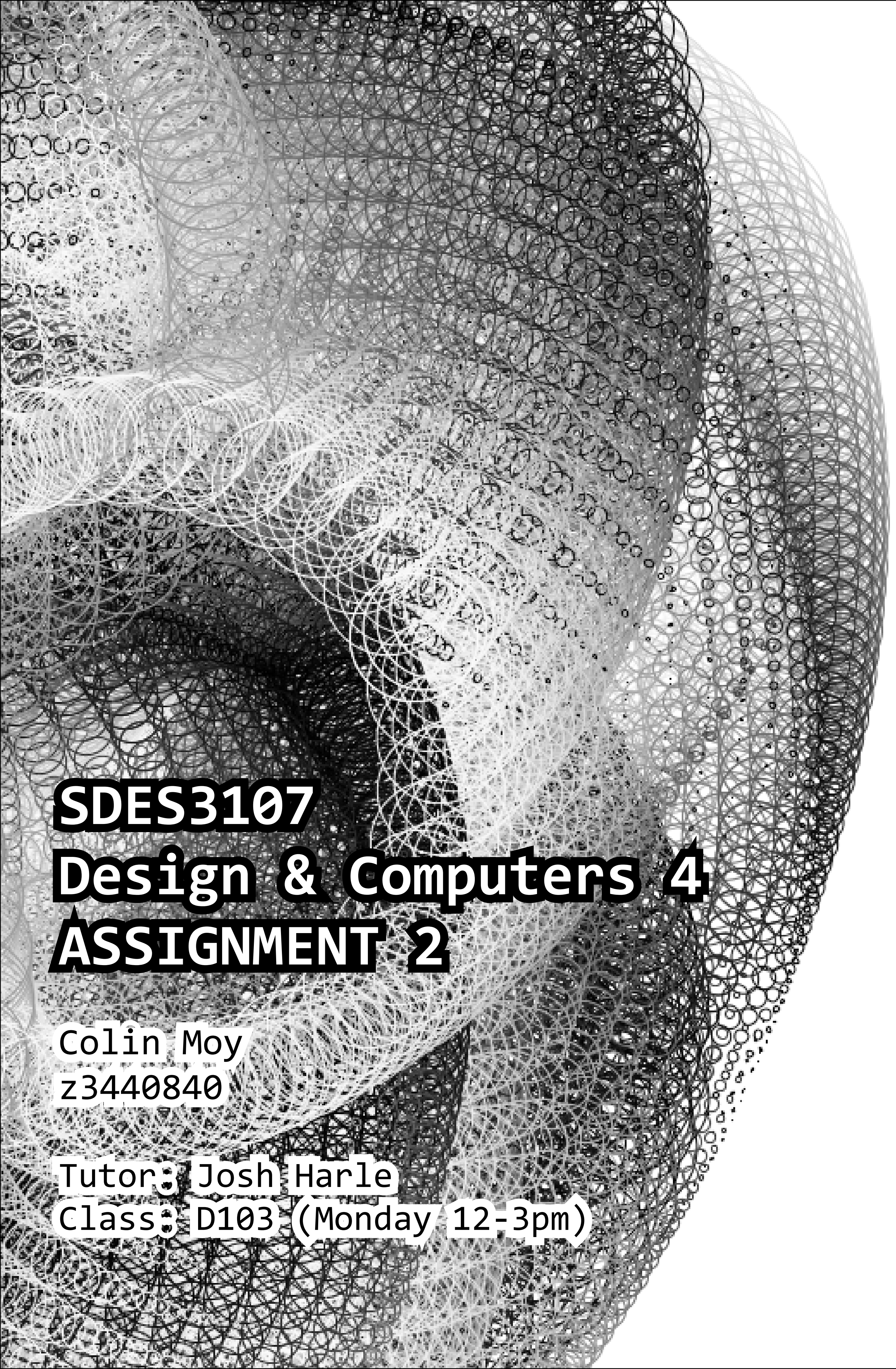
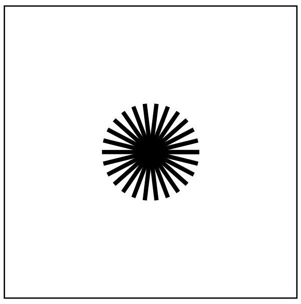


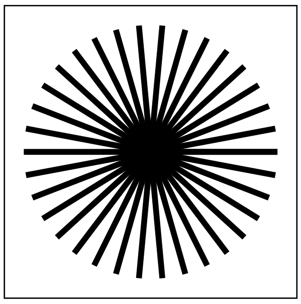
SDES3107
Design & Computers 4
ASSIGNMENT 2

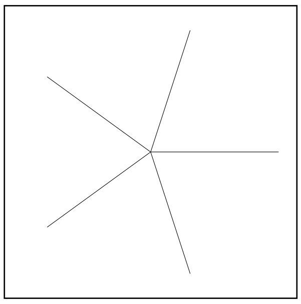
Colin Moy
z3440840

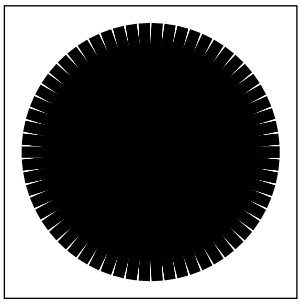
Tutor: Josh Harle
Class: D103 (Monday 12-3pm)

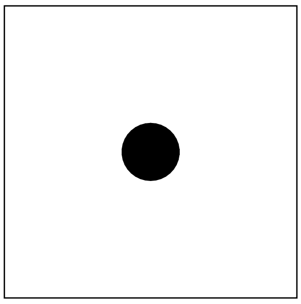
//originalCode: P_2_0_01.pde









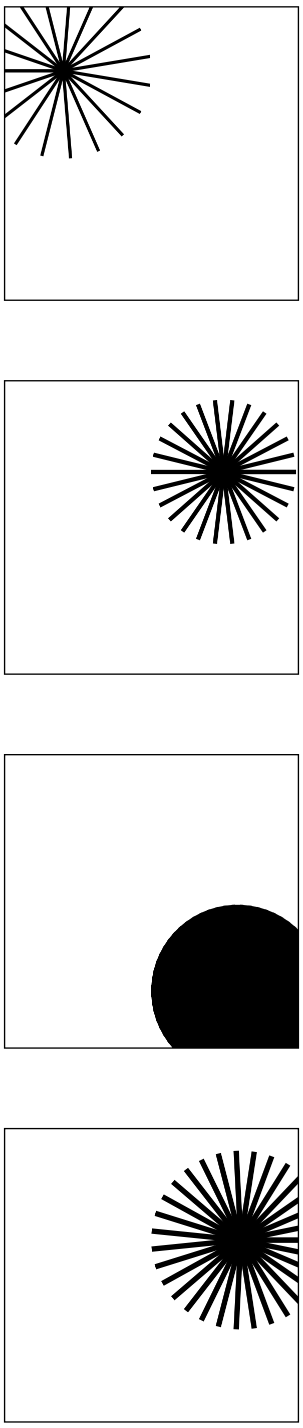


The Original code for this sketch draws lines radiating a circle and the length, thickness and number of lines are deternied by the position of the cursor.

```
// P_2_0_01.pde
//
// Generative Gestaltung, ISBN: 978-3-87439-759-9
// First Edition, Hermann Schmidt, Mainz, 2009
// Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
// Copyright 2009 Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
//
// http://www.generative-gestaltung.de
//
// Licensed under the Apache License, Version 2.0 (the “License”);
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an “AS IS” BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/**
 * drawing a filled circle with lines.
 *
 * MOUSE
 * position x          : length
 * position y          : thickness and number of lines
 *
 * KEYS
 * s                   : save png
 * p                   : save pdf
 */
import java.io.*;
import java.util.*;
import processing.pdf.*;
boolean savePDF = false;

void setup(){
  size(550, 550);
}

void draw(){
  if (savePDF) beginRecord(PDF, timestamp()+”.pdf”);

  strokeCap(SQUARE);
  smooth();
  noFill();
  background(255);
  translate(width/2,height/2);

  int circleResolution = (int) map(mouseY, 0,height, 2,80);
  float radius = mouseX-width/2 + 0.5;
  float angle = TWO_PI/circleResolution;

  strokeWeight(mouseY/20);

  beginShape();
  for (int i=0; i<=circleResolution; i++){
    float x = cos(angle*i) * radius;
    float y = sin(angle*i) * radius;
    line(0, 0, x, y);
    // vertex(x, y);
  }
  endShape();

  if (savePDF) {
    savePDF = false;
    endRecord();
  }
}

void keyPressed() {
  if (key==’s’ || key==’S’) saveFrame(timestamp()+”_##.png”);
  if (key==’p’ || key==’P’) savePDF = true;
}

// timestamp
String timestamp() {
  Calendar now = Calendar.getInstance();
  return String.format(“%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS”, now);
}
```

//CHANGE #1:

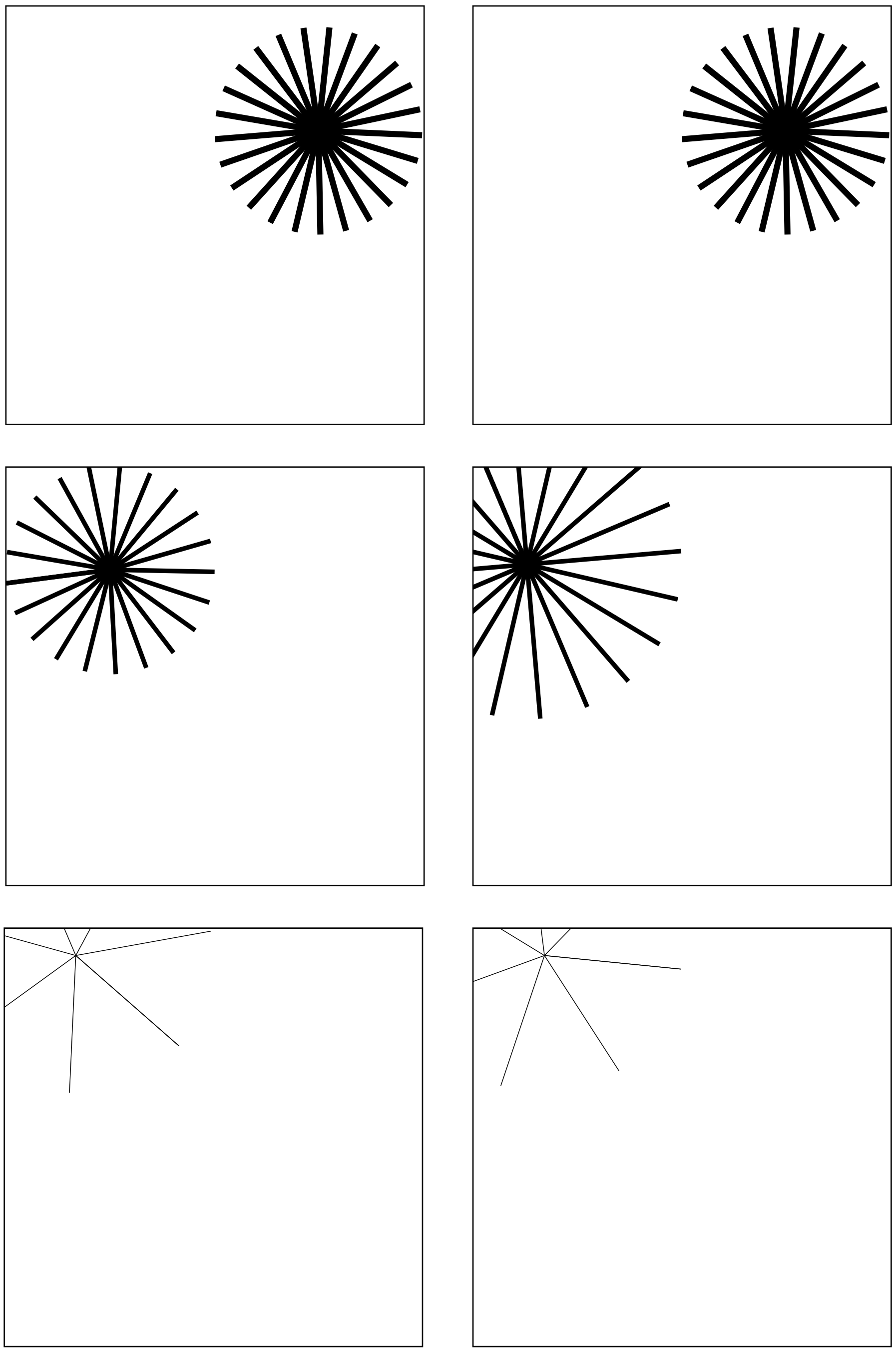
SHAPE FOLLOWS THE MOUSE NOW



```
translate(mouseX, mouseY);
```

//CHANGE #2:

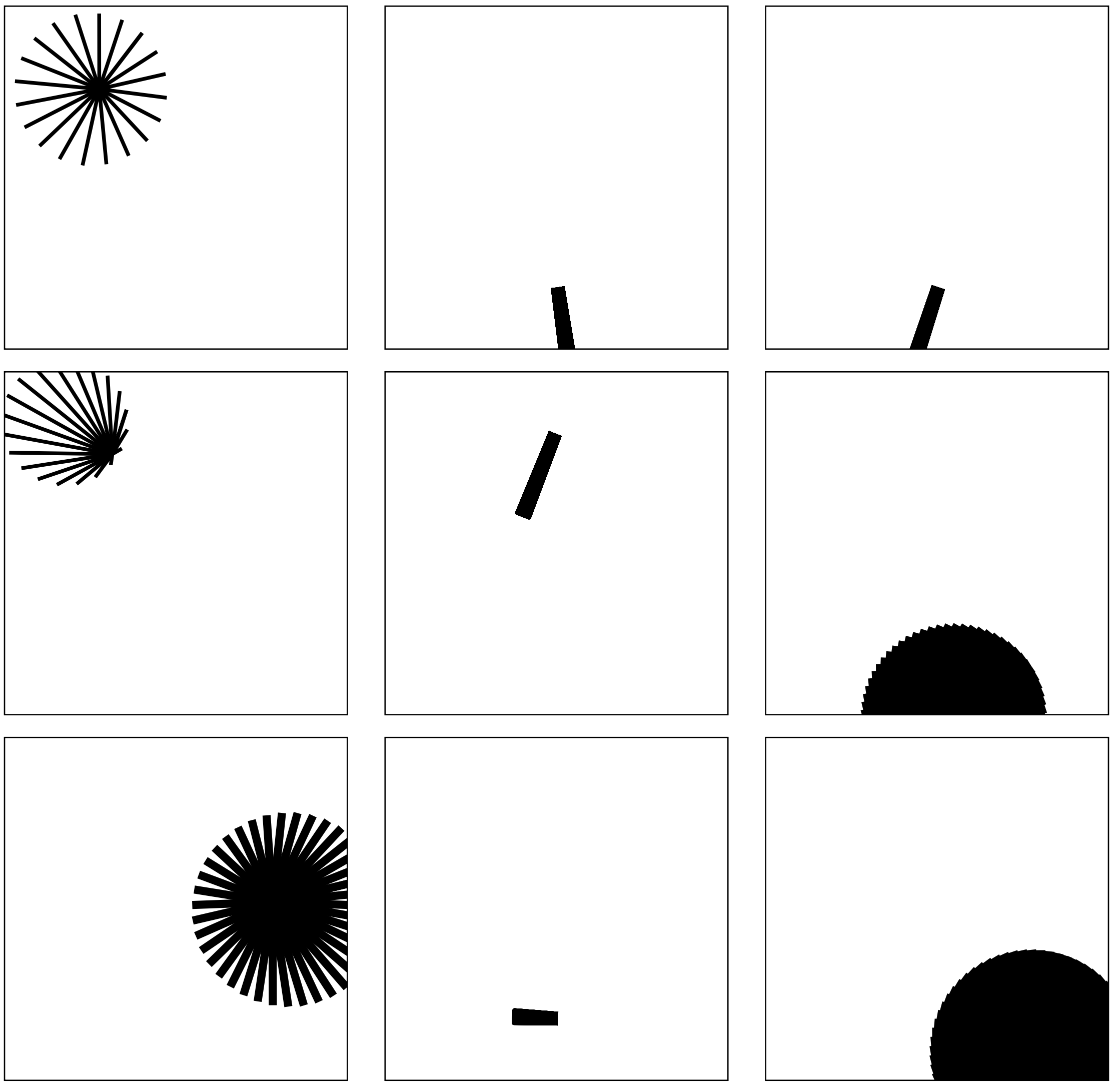
ROTATE SHAPE AROUND CURSOR



```
float rotateDegree = 0;
rotate(rotateDegree/50);
rotateDegree++;
```

//CHANGE #3:

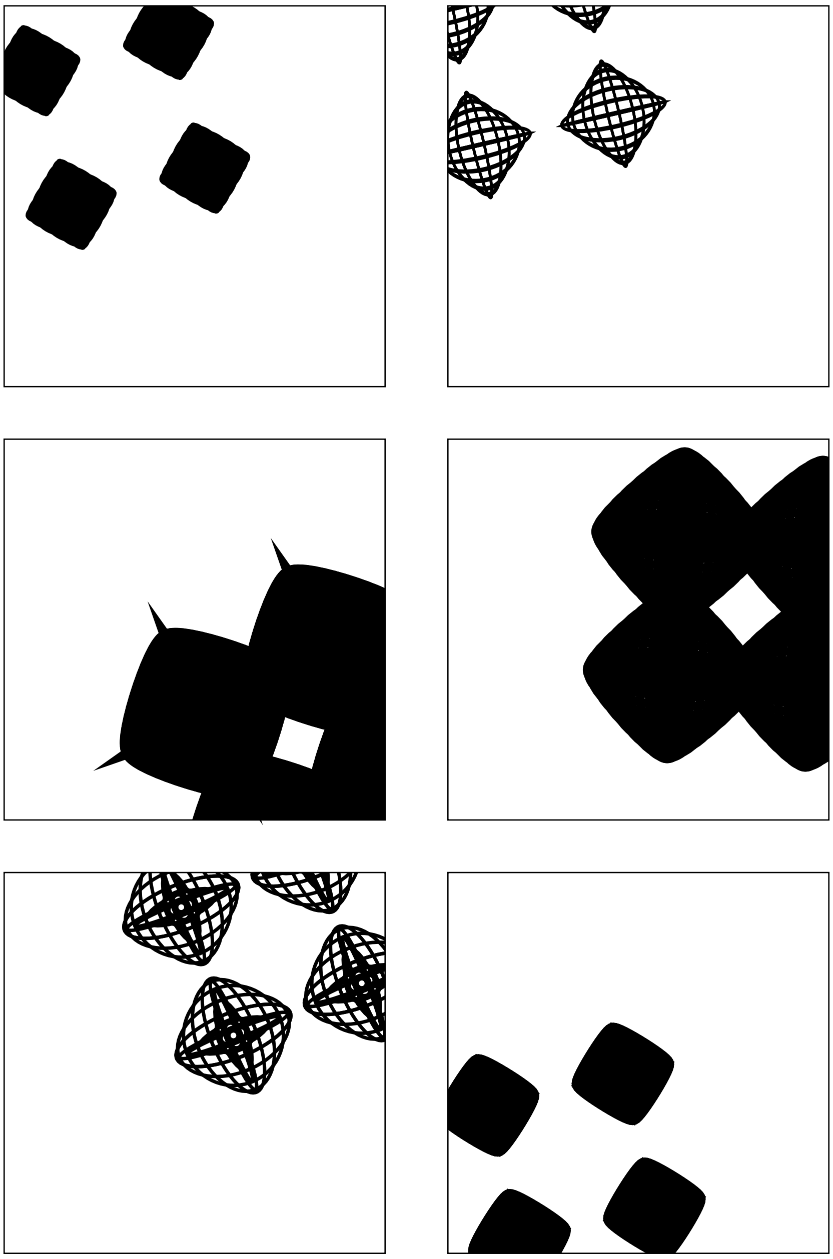
MAKE LINES “BOUNCE OFF” CURSOR WITH A SINE WAVE



```
float waveVal = 0;
waveVal = 50+sin(PI/50*rotateDegree)*50;

beginShape();
  for (int i=0; i<=circleResolution; i++){
    float x = cos(angle*i) * radius;
    float y = sin(angle*i) * radius;
    line(waveVal, waveVal, x, y);
    // vertex(x, y);
  }
  endShape();
```

//CHANGE #4:

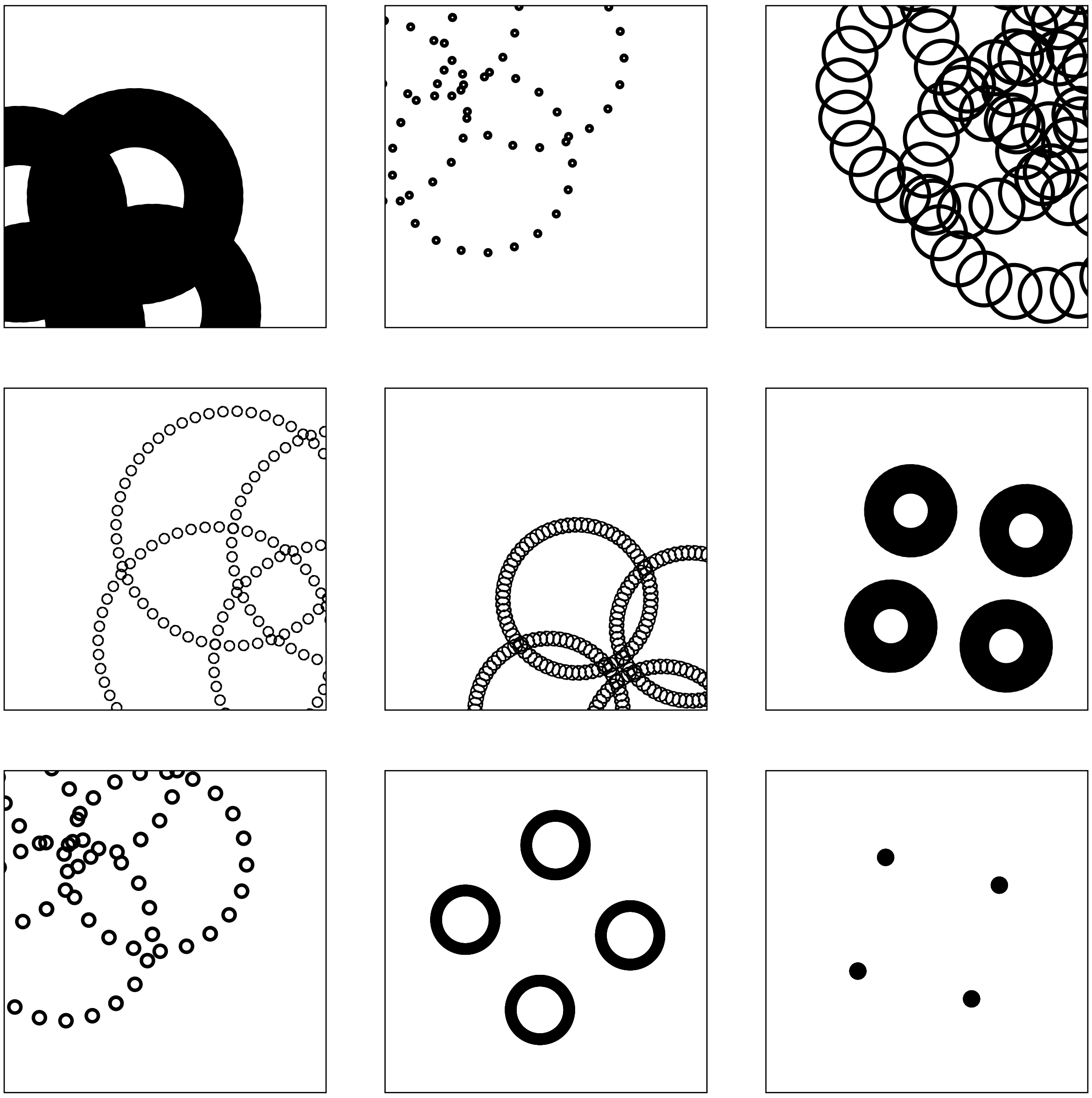
CHANGE THE LINES TO ELLIPSES AND ADD 3 MORE SETS OF ELLIPSES



```
 for (int i=0; i<=circleResolution; i++){
  float x = cos(angle*i) * radius;
  float y = sin(angle*i) * radius;

ellipse(100+waveVal, 100+waveVal, x, y);
ellipse(waveVal-100, waveVal-100, x, y);
ellipse(waveVal-100, 100+waveVal, x, y);
ellipse(100+waveVal, waveVal-100, x, y);

}
```

//CHANGE #5:

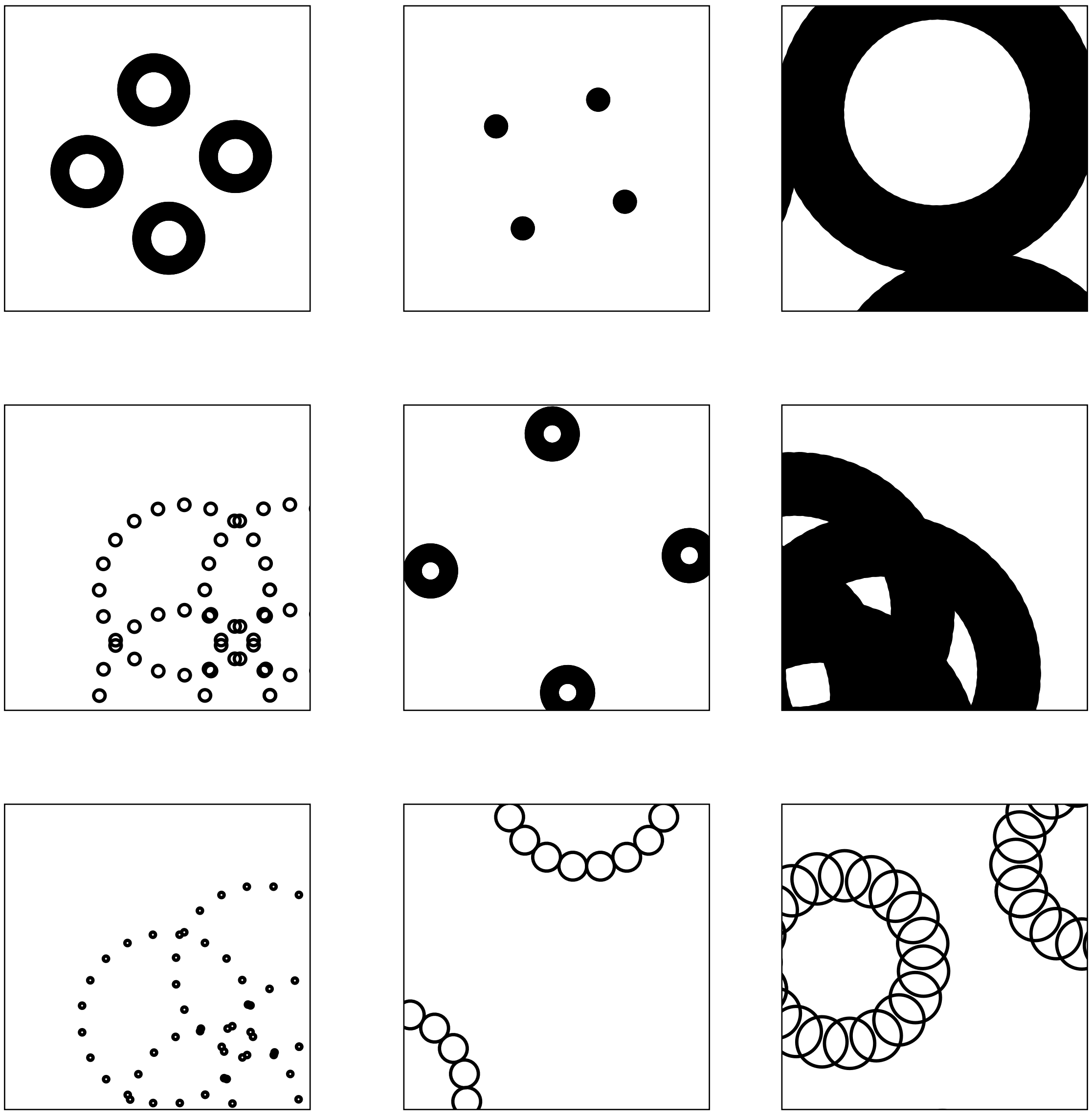
FLUCUATE THE ELLIPSES’ POSITIONS WITH THE CURSOR (FLOAT X AND FLOAT Y) INSTEAD OF THE WIDTH AND HEIGHT OF THE ELLIPSES.



```
 for (int i=0; i<=circleResolution; i++){
  float x = cos(angle*i) * radius;
  float y = sin(angle*i) * radius;

ellipse(100+x, 100+y, waveVal, waveVal);
ellipse(x-100, y-100, waveVal, waveVal);
ellipse(x-100, 100+y, waveVal, waveVal);
ellipse(100+x, y-100, waveVal, waveVal);

}
```

//CHANGE #6:

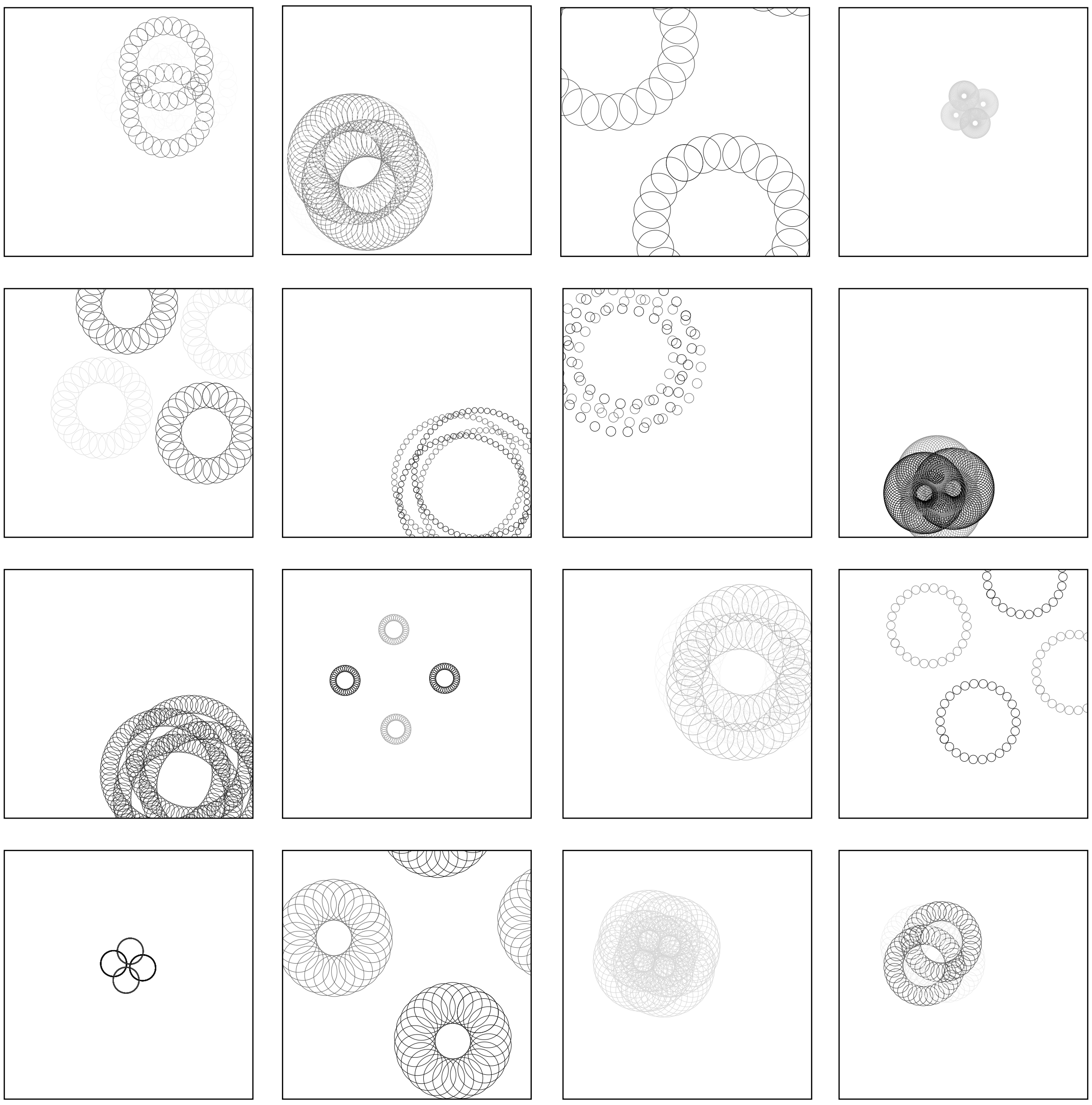
MAKE THE 4 SETS OF SHAPES MOVE APART FROM EACH OTHER WHEN MOUSE IS LEFT CLICKED AND REVERT BACK WHEN MOUSE RELEASED



```
int clickChange = 0;

clickChange = constrain(clickChange,30,300);
if ((mousePressed==true)&&(mouseButton==LEFT))
{clickChange+=5;} else {clickChange-=5;}

ellipse(clickChange+x, clickChange+y, waveVal, waveVal);
ellipse(x-clickChange, y-clickChange, waveVal, waveVal);
ellipse(x-clickChange, clickChange+y,waveVal, waveVal);
ellipse(clickChange+x, y-clickChange, waveVal, waveVal);
```

//CHANGE #7:

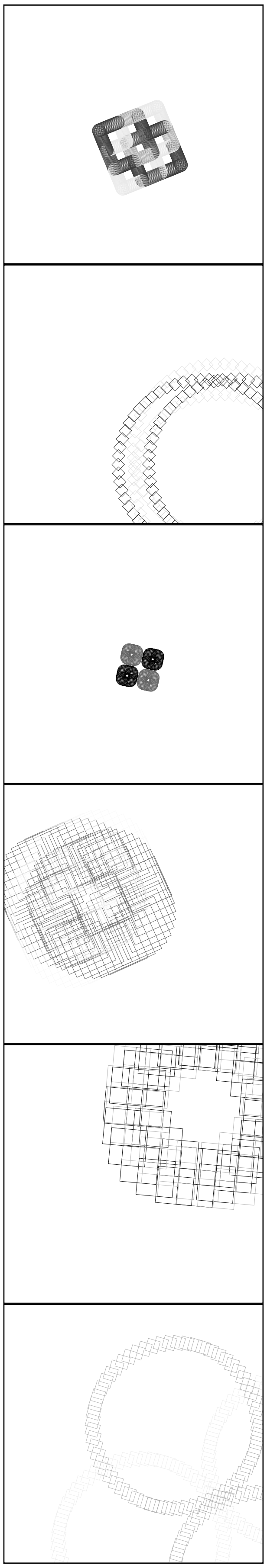
MAKE THE SETS OF SHAPES FADE IN AND OUT 2 BY 2, ONE USING A COSINE WAVE AND THE OTHER A SINE WAVE



```
float fade = 0;
float fade1 = 0;
float fade2 = 0;

stroke(fade1);
ellipse(clickChange+x, clickChange+y, waveVal, waveVal);
ellipse(x-clickChange, y-clickChange, waveVal, waveVal);
stroke(fade2);
ellipse(x-clickChange, clickChange+y,waveVal, waveVal);
ellipse(clickChange+x, y-clickChange, waveVal, waveVal);

fade+=0.1;
fade1=126+sin(radians(fade))*(126);
fade2=126+cos(radians(fade))*(126);
```

//CHANGE #8:

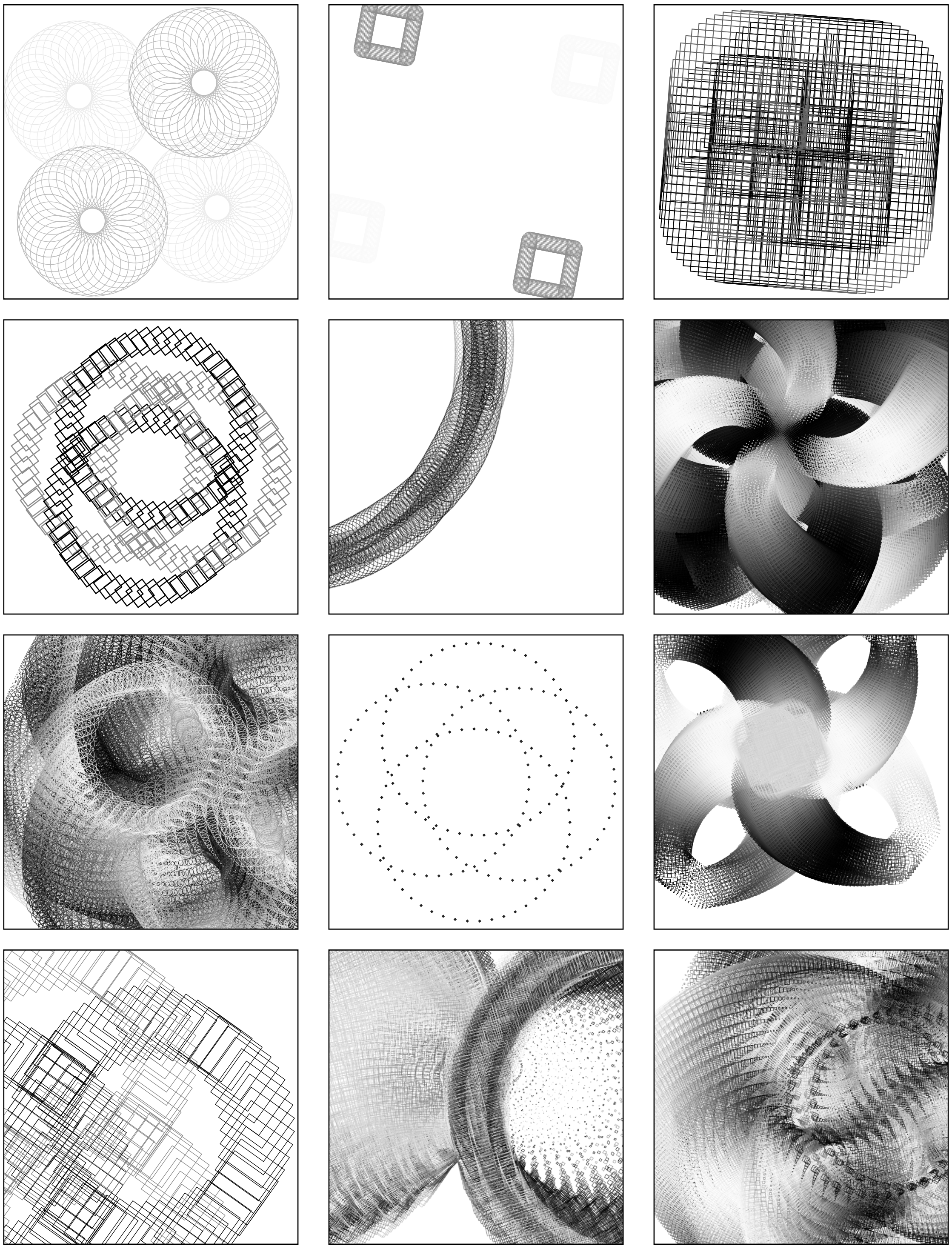
MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”



```
boolean shape = true;
rectMode(CENTER);

if (shape==true){circles();}
if (shape==false){squares();}

void circles()
{
  int circleResolution = (int) map(mouseY, 0, height, 2, 80);
  float radius = mouseX-width/2;
  float angle = TWO_PI/circleResolution;

  waveVal = 50+sin(PI/50*rotateDegree)*50;
  for (int i=0; i<=circleResolution; i++)
    {
      float x = cos(angle*i) * radius;
      float y = sin(angle*i) * radius;

      stroke(fade1);
      ellipse(clickChange+x, clickChange+y, waveVal, waveVal);
      ellipse(x-clickChange, y-clickChange, waveVal, waveVal);

      stroke(fade2);
      ellipse(x-clickChange, clickChange+y,waveVal, waveVal);
      ellipse(clickChange+x, y-clickChange, waveVal, waveVal);

      fade+=0.1;
      fade1=126+sin(radians(fade))*(126);
      fade2=126+cos(radians(fade))*(126);
    }
}

void squares()
{
  int squareResolution = (int) map(mouseY, 0, height, 2, 80);
  float radius = mouseX-width/2;
  float angle = TWO_PI/squareResolution;

  waveVal = 50+sin(PI/50*rotateDegree)*50;

  for (int i=0; i<=squareResolution; i++)
    {
      float x = cos(angle*i) * radius;
      float y = sin(angle*i) * radius;
      stroke(fade1);
      rect(clickChange+x, clickChange+y, waveVal, waveVal);
      rect(x-clickChange, y-clickChange, waveVal, waveVal);
      stroke(fade2);
      rect(x-clickChange, clickChange+y,waveVal, waveVal);
      rect(clickChange+x, y-clickChange, waveVal, waveVal);

      fade+=0.1;
      fade1=126+sin(radians(fade))*(126);
      fade2=126+cos(radians(fade))*(126);
    }
}
```

//CHANGE #9:

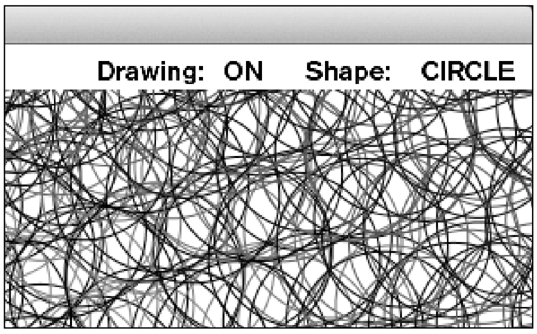
WHEN THE RIGHT MOUSE BUTTON IS CLICKED, DRAWING WITH THE SHAPE IS TURNED ON, CLICK AGAIN TO TURN OFF

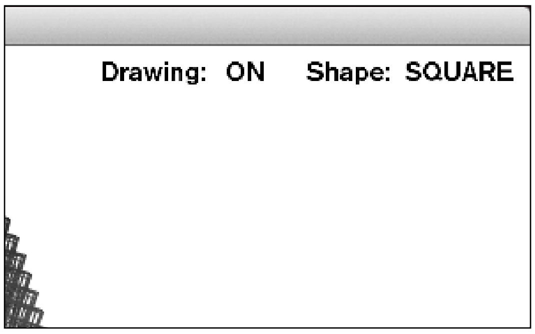


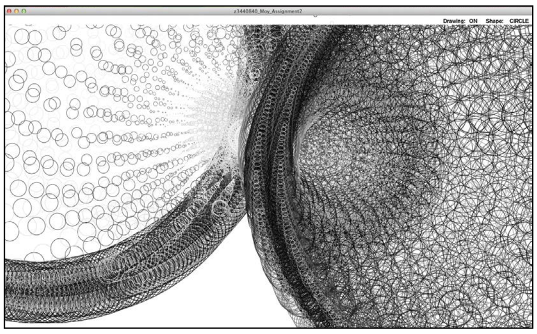
```
boolean drawing = false;

void mouseReleased()
{
 if (mouseButton==RIGHT)
 {drawing=!drawing; background(255);}
}
```

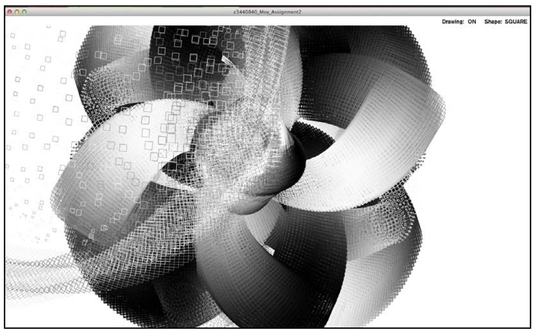
//CHANGE #10:

ADDED A DISPLAY BAR THAT TELLS YOU IF DRAWING IS ON OR OFF AND WHICH SHAPE YOU ARE WORKING WITH AND ADJUSTED SKETCH TO FULLSCREEN.


Drawing: ON Shape: CIRCLE

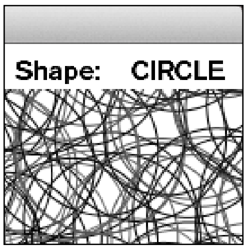

Drawing: ON Shape: SQUARE

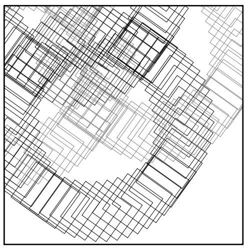

Drawing: ON Shape: CIRCLE

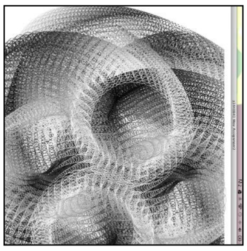

Drawing: ON Shape: SQUARE

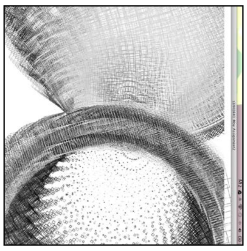
```
size(displayWidth,displayHeight);

textAlign(RIGHT);
  PFont font;
  font = loadFont(“AkzidenzGroteskBQ-XBdCndIt-48.vlw”);
  textFont(font);
  textSize(15);

displayBar();

void displayBar()
{
  fill(255);
  noStroke();
  rect(width/2, 10, width,30);

  fill(0);
  if (drawing==false)
  {
    background(255);
    text(“OFF”,(width-150),20);
  }
  if (drawing==true) {text(“ON”,(width-150),20);}

  if (shape==false){text(“SQUARE”,(width-10),20);}
  else {text(“CIRCLE”,(width-10),20);}

  text(“Drawing:              Shape:                  “,width-15, 20);
}
```

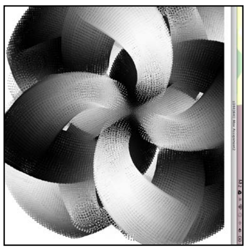
//modifiedCode: z3440840_Moy_Assignment_2

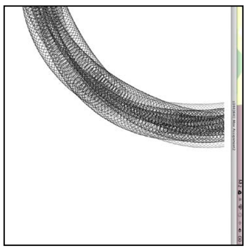


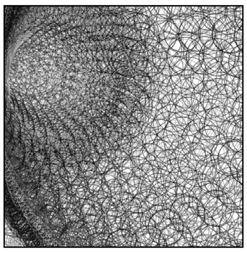












```
// P_2_0_01.pde (EDITTED VERSION BY COLIN MOY)
//LICENSE AND COPYRIGHT OF EDITTED FILE BELOW
//

/*
 * THIS EDITTED VERSION OF SKETCH P_2_0_01
 * DRAWS A ROTATING RING OF 4 RINGS OF SHAPES
 * WITH [[NEW]] CONTROLS THAT HELP YOU ALTER THE APPEARANCE OF THE SHAPE
 *
 * MOUSE
 * Position X          : Distance between shapes within each ring of shapes
 * Position Y          : Number of shapes within each ring of shapes
 * Left Click          : Push the 4 rings of shapes away from the cursor
 * Right Click         : Toggle drawing ON and OFF
 *
 * KEYS
 * Spacebar            : Toggle shapes to be CIRCLES or SQUARES
 * s                   : save png
 * p                   : save pdf
 *
*/


import java.io.*;
import java.util.*;
import processing.pdf.*;

boolean savePDF = false;
boolean shape = true;     // CHANGE #8 (1 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”
boolean drawing = false;  // CHANGE #9 (1 of 2): WHEN THE RIGHT MOUSE BUTTON IS CLICKED, DRAWING WITH THE SHAPE IS TURNED ON, CLICK AGAIN TO TURN OFF

int clickChange = 0;      // CHANGE #6 (1 of 4): MAKE THE 4 SHAPES MOVE APART FROM EACH OTHER WHEN MOUSE IS LEFT CLICKED AND REVERT BACK WHEN MOUSE RELEASED

float rotateDegree = 0;   // CHANGE #2 (1 of 2): ROTATE SHAPE

float fade = 0;           //  \
float fade1 = 0;          //   > CHANGE #7 (1 of 4): MAKE THE SHAPE FADE IN AND OUT 2 BY 2, ONE USING A COSINE WAVE AND THE OTHER A SINE WAVE SO THEY’LL BE “OPPOSITES”
float fade2 = 0;          //  /

float waveVal = 0;        // CHANGE #3 (1 of 4): MAKE LINES “BOUNCE OFF” CURSOR WITH A SINE WAVE

void setup()
{
  size(displayWidth,displayHeight);    // CHANGE #10 (1 of 4): ADDED A DISPLAY BAR THAT TELLS YOU IF DRAWING IS ON OR OFF AND WHICH SHAPE YOU ARE WORKING WITH AND ADJUSTED SKETCH TO FULLSCREEN

  textAlign(RIGHT);                                   //  \
  PFont font;                                         //   \
  font = loadFont(“AkzidenzGroteskBQ-XBdCndIt-48.vlw”);  //    > CHANGE #10 (2 of 4): ADDED A DISPLAY BAR THAT TELLS YOU IF DRAWING IS ON OR OFF AND WHICH SHAPE YOU ARE WORKING WITH AND ADJUSTED SKETCH TO FULLSCREEN
  textFont(font);                                     //   /
  textSize(15);                                       //  /

  smooth();
  ellipseMode(CENTER);  // CHANGE #4 (1 of 3): CHANGE THE LINES TO ELLIPSES AND ADD 3 ELLIPSES
  rectMode(CENTER);     // CHANGE #8 (2 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”

}

void draw()
{ displayBar();         // CHANGE #10 (3 of 4): ADDED A DISPLAY BAR THAT TELLS YOU IF DRAWING IS ON OR OFF AND WHICH SHAPE YOU ARE WORKING WITH AND ADJUSTED SKETCH TO FULLSCREEN

  noFill();
  if (savePDF) beginRecord(PDF, timestamp()+”.pdf”);

  translate(mouseX, mouseY);     // CHANGE #1 (1 of 1): SHAPE FOLLOWS THE MOUSE NOW
  rotate(rotateDegree/50);       //   \
  rotateDegree++;                //   / CHANGE #2 (2 of 2): ROTATE SHAPE

  if (shape==true){circles();}   //   \
  if (shape==false){squares();}  //   / CHANGE #8 (3 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”

  if (savePDF){savePDF = false; endRecord();}

  clickChange = constrain(clickChange,30,300);    //   \
  if ((mousePressed==true)&&(mouseButton==LEFT))  //    > CHANGE #6 (2 of 4): MAKE THE 4 SHAPES MOVE APART FROM EACH OTHER WHEN MOUSE IS LEFT CLICKED AND REVERT BACK WHEN MOUSE RELEASED
  {clickChange+=5;} else {clickChange-=5;}        //   /


}



void circles()    // CHANGE #8 (4 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”
{
  int circleResolution = (int) map(mouseY, 0, height, 2, 80);
  float radius = mouseX-width/2;
  float angle = TWO_PI/circleResolution;

  waveVal = 50+sin(PI/50*rotateDegree)*50;  // CHANGE #3 (2 of 4): MAKE LINES “BOUNCE OFF” CURSOR WITH A SINE WAVE

  for (int i=0; i<=circleResolution; i++)
    {
      float x = cos(angle*i) * radius;
      float y = sin(angle*i) * radius;

      stroke(fade1);//CHANGE #7 (2 of 4): MAKE THE SHAPE FADE IN AND OUT 2 BY 2, ONE USING A COSINE WAVE AND THE OTHER A SINE WAVE SO THEY’LL BE “OPPOSITES”
      ellipse(clickChange+x, clickChange+y, waveVal, waveVal);  //  \ CHANGE #3 (3 of 4): MAKE LINES “BOUNCE OFF” CURSOR WITH A SINE WAVE
      ellipse(x-clickChange, y-clickChange, waveVal, waveVal);  //  / CHANGE #4 (2 of 3): CHANGE THE LINES TO ELLIPSES AND ADD 3 ELLIPSES
      //CHANGE #6 (3 of 4): MAKE THE 4 SHAPES MOVE APART FROM EACH OTHER WHEN MOUSE IS LEFT CLICKED AND REVERT BACK WHEN MOUSE RELEASED

      stroke(fade2);//CHANGE #7 (3 of 4): MAKE THE SHAPE FADE IN AND OUT 2 BY 2, ONE USING A COSINE WAVE AND THE OTHER A SINE WAVE SO THEY’LL BE “OPPOSITES”
      ellipse(x-clickChange, clickChange+y,waveVal, waveVal);   //  \ CHANGE #3 (4 of 4): MAKE LINES “BOUNCE OFF” CURSOR WITH A SINE WAVE
      ellipse(clickChange+x, y-clickChange, waveVal, waveVal);  //  / CHANGE #4 (3 of 3): CHANGE THE LINES TO ELLIPSES AND ADD 3 ELLIPSES
      // CHANGE #5 (1 of 1): FLUCUATE THE ELLIPSES’ POSITIONS WITH THE CURSOR (FLOAT X AND FLOAT Y) INSTEAD OF THE WIDTH AND HEIGHT OF THE ELLIPSES.
      //CHANGE #6 (4 of 4): MAKE THE 4 SHAPES MOVE APART FROM EACH OTHER WHEN MOUSE IS LEFT CLICKED AND REVERT BACK WHEN MOUSE RELEASED

      fade+=0.1;                           //   \
      fade1=126+sin(radians(fade))*(126);  //    > CHANGE #7 (4 of 4): MAKE THE SHAPE FADE IN AND OUT 2 BY 2, ONE USING A COSINE WAVE AND THE OTHER A SINE WAVE SO THEY’LL BE “OPPOSITES”
      fade2=126+cos(radians(fade))*(126);  //   /
    }
}

void squares()                                                      //   \
{                                                                   //    \
  int squareResolution = (int) map(mouseY, 0, height, 2, 80);       //     \
  float radius = mouseX-width/2;                                   //      \
  float angle = TWO_PI/squareResolution;                           //       \
                                                                    //        \
  waveVal = 50+sin(PI/50*rotateDegree)*50;                          //         \
                                                                    //          \
  for (int i=0; i<=squareResolution; i++)                           //           \
    {                                                               //            \
      float x = cos(angle*i) * radius;                             //             \
      float y = sin(angle*i) * radius;                             //              > CHANGE #8 (5 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”
      stroke(fade1);                                                //             /
      rect(clickChange+x, clickChange+y, waveVal, waveVal);         //            /
      rect(x-clickChange, y-clickChange, waveVal, waveVal);         //           /
      stroke(fade2);                                                //          /
      rect(x-clickChange, clickChange+y,waveVal, waveVal);          //         /
      rect(clickChange+x, y-clickChange, waveVal, waveVal);         //        /
                                                                    //       /
      fade+=0.1;                                                    //      /
      fade1=126+sin(radians(fade))*(126);                           //     /
      fade2=126+cos(radians(fade))*(126);                           //    /
    }
}

void displayBar()                                                     // \
{                                                                     //  \
  fill(255);                                                         //   \
  noStroke();                                                         //    \
  rect(width/2, 10, width,30);                                        //     \
                                                                      //      \
  fill(0);                                                           //       \
  if (drawing==false)                                                 //        \
  {                                                                   //         \
    background(255);                                                  //          > CHANGE #10 (4 of 4): ADDED A DISPLAY BAR THAT TELLS YOU IF DRAWING IS ON OR OFF AND WHICH SHAPE YOU ARE WORKING WITH AND ADJUSTED SKETCH TO FULLSCREEN
    text(“OFF”,(width-150),20);                                       //         /
  }                                                                   //        /
  if (drawing==true) {text(“ON”,(width-150),20);}                     //       /
                                                                      //      /
  if (shape==false){text(“SQUARE”,(width-10),20);}                    //     /
  else {text(“CIRCLE”,(width-10),20);}                                //    /
                                                                      //   /
  text(“Drawing:              Shape:                “,width-15, 20);  //  /
}                                                                     // /

void mouseReleased()                    //  \
{                                       //   \
 if (mouseButton==RIGHT)                //    > // CHANGE #9 (2 of 2): WHEN THE RIGHT MOUSE BUTTON IS CLICKED, DRAWING WITH THE SHAPE IS TURNED ON, CLICK AGAIN TO TURN OFF
 {drawing=!drawing; background(255);}   //   /
}                                       //  /

void keyPressed()
{
  if (key==’s’ || key==’S’) saveFrame(timestamp()+”_##.png”);
  if (key==’p’ || key==’P’) savePDF = true;
}

void keyReleased()                  //  \
{if (key==’ ‘) {shape=!shape;}}     //  / CHANGE #8 (6 of 6): MAKE THE SHAPE CHANGE WHEN SPACEBAR IS TAPPED, BY USING 2 DIFFERENT FUNCTIONS “CIRCLES” AND “SQUARES”

// timestamp
String timestamp()
{
  Calendar now = Calendar.getInstance();
  return String.format(“%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS”, now);
}



//LICENSE AND COPYRIGHT OF EDITTED FILE
//
// Generative Gestaltung, ISBN: 978-3-87439-759-9
// First Edition, Hermann Schmidt, Mainz, 2009
// Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
// Copyright 2009 Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
//
// http://www.generative-gestaltung.de
//
// Licensed under the Apache License, Version 2.0 (the “License”);
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an “AS IS” BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
```