Old 257 Exam #2s for Practice

Exams will be taken on Thursday March 27 in the cluster. You will have the entire class time to do the exam. If you finish early, you may leave or stay and work. If you do not finish by 1:20, you will not receive extra time.

The exam will consist of four problems in which you write code. You will need to create the folders and files required. You must be able to submit them to the handin folder created for you in the course directory.

The problems will focus on the fundamentals of programming with Processing that were presented in homeworks 1-11. Homework 12 and any stuff on PImage, String, and other stuff is not on the exam.

You will have access to the Processing API. You MAY NOT access any previous code you or anyone else has written.

We will, where possible help you with syntax errors but we will not help you with logic problems.

There will be no new algorithmic problems on the exam. Everything you will be asked to do is a version of what was written class or a version of what you were asked to write for a homework assignment.

Improperly working code can receive partial credit but it must compile. Code that fails to compile receives a zero.

Remember that the order of topics and homeworks can vary from semester so some of the questions on these old exams may not be pertinent to the material covered in this term's homeworks 1-8. Strings will NOT be on the exam.

Version 0, the first sample exam is the most recent exam. It was used this past fall. The last sample exam is the oldest exam. The order of topics vary from term to term so some of the questions are not appropriate for this semester's Exam 2.

Old Exam 2 Version 0 Question 1 - 10 points

Copy the following code into a processing program:

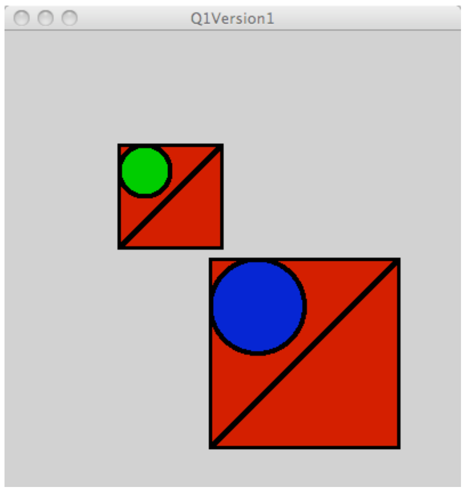
```
color c1, c2, c3;

void setup( )
{
  size( 400, 400 );
  c1 = color( 200, 0, 0 );   // red
  c2 = color( 0, 200, 0 );   // green
  c3 = color( 0, 0, 200 );   // blue
  strokeWeight( 3 );
  ellipseMode( CORNER );
}

void draw( )
{
   background( 200 );
   design( 100, 100,  90, c1, c2 );   // smaller design
   design( 180, 200, 165, c1, c3 );   // larger design
   noLoop( );
}
```

Define the function `design( float, float, float, color, color)`
where the arguments are:

- argument #1 is the x coordinate of the rectangle
- argument #2 is the y coordinate of the rectangle
- argument #3 is the size of the rectangle
- argument #4 is the fill color of the rectangle
- argument #5 is the fill color of the circle

The mode and stroke weight are set properly in the `setup( )` function.

Expected output:



Question 2 - 10 points

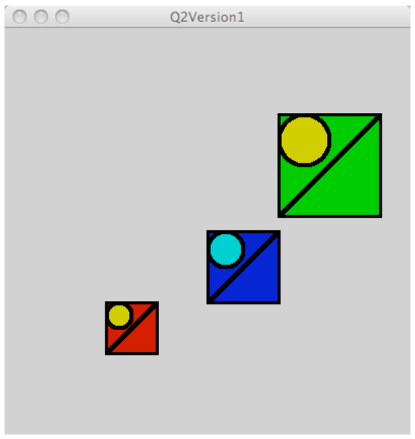
Open a new, empty Processing program.

- Copy the following code into the program:

```
int    [ ] xCoords    =  { 100, 200,  270 };
int    [ ] yCoords    =  { 270, 200,   85 };
int    [ ] dimensions =  {  50,  70,  100 };

color [ ] colorListRect =  { color( 200, 0, 0 ),   // rect color
                              color( 0, 0, 200 ),
                              color( 0, 200, 0 )
                            };
color [ ] colorListEllipse = {  color( 200, 200, 0 ), // ellipse color
                                 color( 0, 200, 200 ),
                                 color( 200, 200, 0 )
                              };
void setup( )
{
  size( 400, 400 );
  strokeWeight( 3 );
  ellipseMode( CORNER );
}

void draw( )
{
   background( 200 );
   drawAllDesigns(  );
   noLoop( );
}
```

- Copy your `design( )` function from question 1 into the program.
- Define the function, `drawAllDesigns( )`. The `drawAllDesigns( )` function must use a loop to traverse the parallel arrays declared above and use the values in the arrays to draw the three designs. The function must call `design( )` inside the loop.
- You MAY NOT hardwire the loop to work with only three-element arrays. Future arrays may be longer or shorter. Your code must work with parallel arrays of any length.

Expected output:



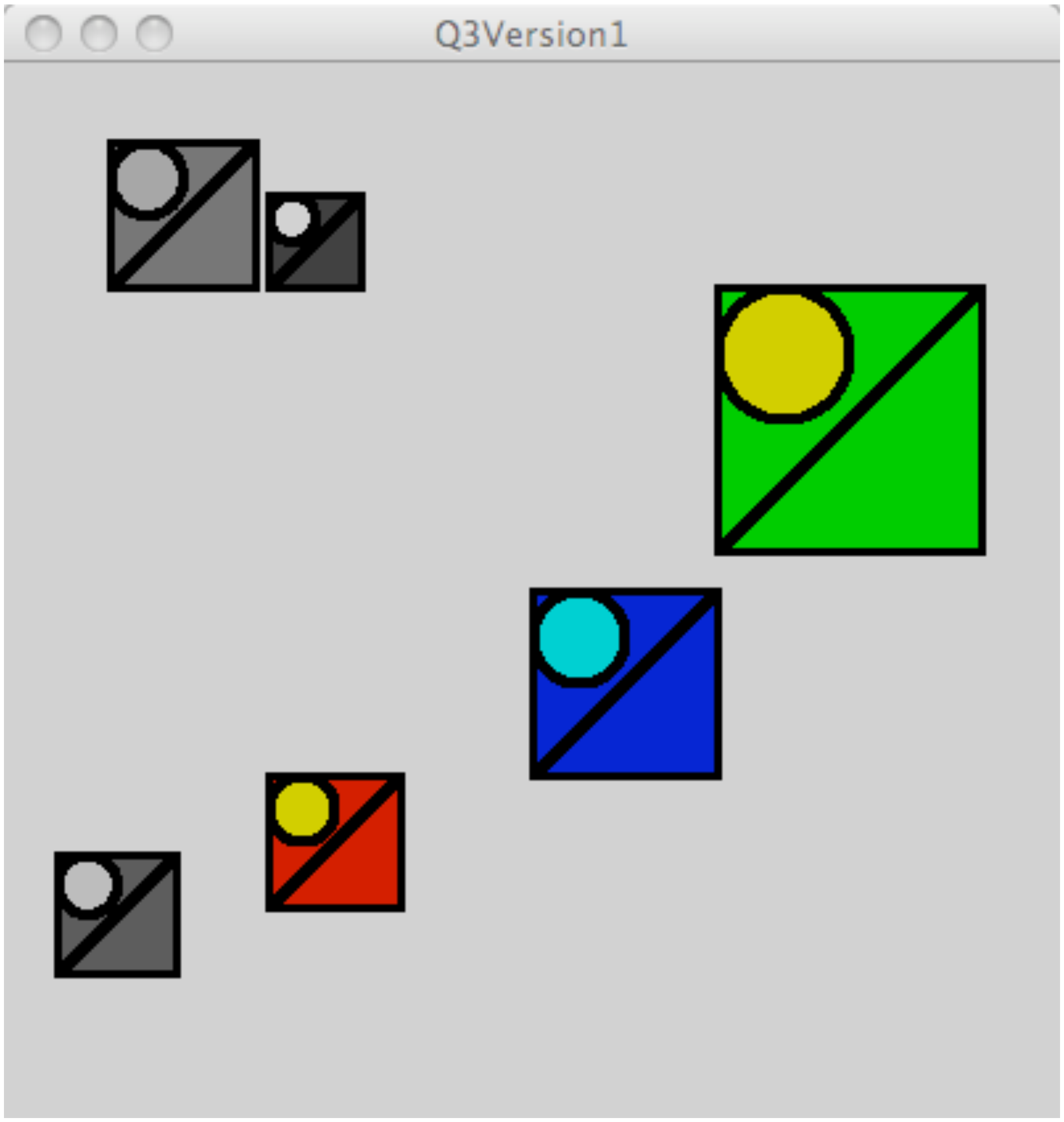
Question 3 - 15 points

Open a new, empty Processing program.
- Copy the following code into the program:

```
int   [ ] xCoords1    =  { 100, 200,  270 };
int   [ ] yCoords1    =  { 270, 200,   85 };
int   [ ] dimensions1 =  {  50,  70,  100 };

color [ ] colorListRect1 =  { color( 200, 0, 0 ),
                              color( 0, 0, 200 ),
                              color( 0, 200, 0 )
                            };
color [ ] colorListEllipse1 = {  color( 200, 200, 0 ),
                                 color( 0, 200, 200 ),
                                 color( 200, 200, 0 )
                              };

int   [ ] xCoords2    =  { 100,  20, 40 };
int   [ ] yCoords2    =  {  50, 300, 30 };
int   [ ] dimensions2 =  {  35,  45, 55 };

color [ ] colorListRect2 =  { color( 50 ),
                              color( 75 ),
                              color( 100 )
                            };
color [ ] colorListEllipse2 = {  color( 200 ),
                                 color( 175 ),
                                 color( 150 )
                              };

void setup( )
{
  size( 400, 400 );
  strokeWeight( 3 );
  ellipseMode( CORNER );
}

void draw( )
{
   background( 200 );
   drawAllDesigns( xCoords1, yCoords1, dimensions1,
                   colorListRect1, colorListEllipse1 );
   drawAllDesigns( xCoords2, yCoords2, dimensions2,
                   colorListRect2, colorListEllipse2 );
   noLoop( );
}
```

- Copy your `design( )` function from question 1 into the program.
- Copy your `drawAllDesigns( )` from question 2 into the program.
- Modify your `drawAllDesigns( )`function to accept arguments and so it can traverse different arrays to draw the design.
- You may not write two different functions do to this. One definition of `drawAllDesigns( )`must be able to traverse any one of the two sets of arrays.

Expected outputs are on the following page:


Q3Version1

// Question 4 - 15 points

// Copy the following code into a new program:

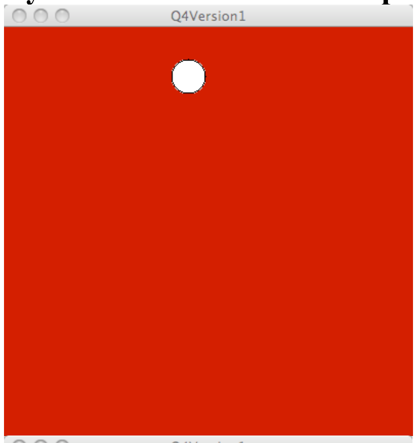
```
int x, y, diam, deltaX, deltaY;
color upperThird, middleThird, lowerThird;
color backgroundColor;
void setup( )
{
   size( 400, 400 );
   x = 0;
   y = 0;
   deltaX = 30;
   deltaY = 3;
   diam = 33;
   upperThird= color( 200, 0, 0 );
   middleThird = color( 0, 200, 0 );
   lowerThird = color ( 0, 0, 200);
   backgroundColor = upperThird;
   noStroke( );
}

void draw( )
{
   setBackgroundColor( );
   ellipse( x, y, diam, diam );
   moveEllipse( );
}

void moveEllipse( )
{
    y += deltaY;
    if ( y > height )
    {
       y = 0;
       x += deltaX;
       if (x > width)
         x = 0;
    }
}

void setBackgroundColor(   )
{

}
```

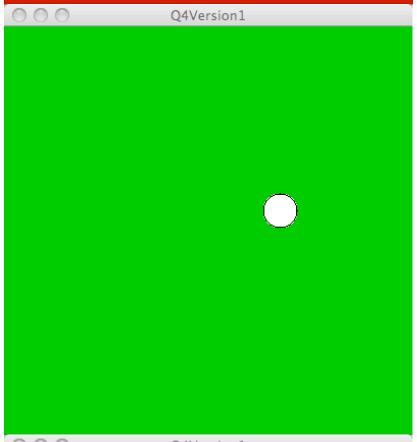
Finish the definition of `setBackgroundColor(   )`.
The function sets the color based on the vertical location of the ellipse.

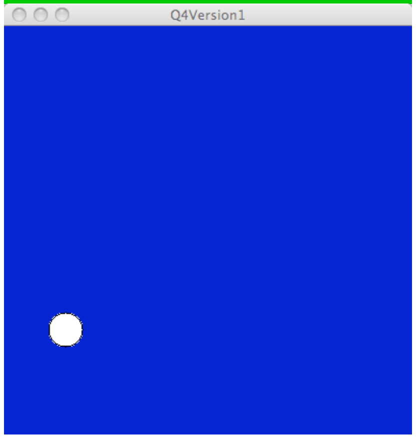
- upper one third of the window – the ellipse is red
- middle one third of the window – the ellipse is green

lower one third of the window – the ellipse is blue

Sample output – the three images below are taken at different times during the run of the program. The primary movement of the white sphere is vertical:







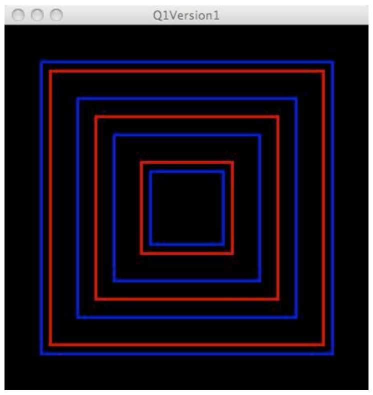
Old Exam 2 Version 1 Question 1 - 10 points

Copy the following code into a processing program:

```
color c1, c2;

void setup( )
{
  size( 400, 400 );
  c1 = color( 200, 0, 0 );  // red
  c2 = color( 0, 0, 200 );  // blue
  strokeWeight( 3 );
  noFill( );
  rectMode( CENTER );
}

void draw( )
{
   background( 0 );
   design( 100, c1 );   // red  rects with 100 pixel separation
   design( 80, c2 );    // blue rects with  80 pixel separation
   noLoop( );
}
```

Define the function `design( float, color)`
where the arguments are:
argument #1 width and height of the innermost rectangle. It is also the amount added to the width and height of each subsequent rectangle drawn.
argument #2 is the color of the stroke making the rectangle.
Rectangles are drawn as long as they are visible in the window.
The fill value, strokeWeight value, and rectMode are set correctly in the setup( ) function.

You will need to use some form of loop to do this. Hardwiring the number of rectangles will not receive full credit.
Expected output:



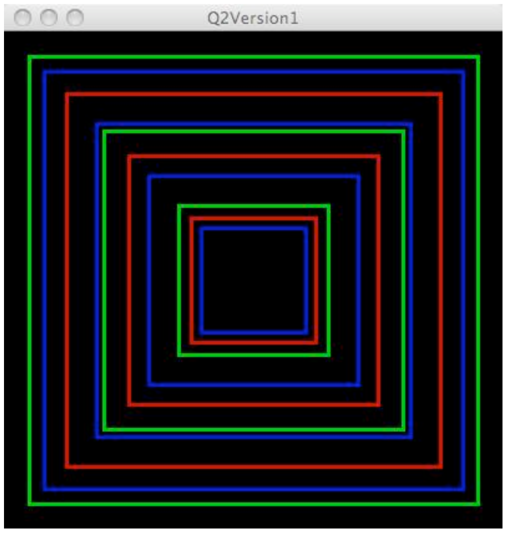
Question 2 - 10 points

Open a new, empty Processing program.
- Copy the following code into the program:

```
int   [ ] spacingList =  { 100, 84, 120 };
color [ ] colorList   =  {  color( 200, 0, 0 ),
                            color( 0, 0, 200 ),
                            color( 0, 200, 0 ) };
void setup( )
{
  size( 400, 400 );
  strokeWeight( 3 );
  noFill( );
  rectMode( CENTER );
}

void draw( )
{
   background( 0 );
   drawAllDesigns( );
   noLoop( );
}
```

- Copy your `design( )` function from question 1 into the program.
- Define the function, `drawAllDesigns( )` so that it traverses the arrays and draws designs using the data in the arrays.
- Your `drawAllDesigns( )` must use the `design( )` function you defined in question 1 to receive full credit for this problem. If you were not able to get question one working, you can put code to draw the rectangles inside the loop that is traversing the arrays for partial credit.
- You MAY NOT hardwire the loop to work with only three-element arrays.
- Your code must work with parallel arrays of any length.

Expected output:



Question 3 - 15 points

Open a new, empty Processing program.
- Copy the following code into the program:

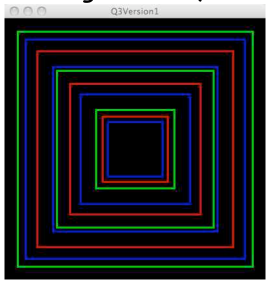
```
int   [ ] spacingList1 = { 100,  84, 120 };
int   [ ] spacingList2 = { 120, 140, 170 };
int   [ ] spacingList3=  { 130, 170, 190 };
color [ ] colorList   =  { color( 200, 0, 0 ),
                           color( 0, 0, 200 ),
                           color( 0, 200, 0 ) };

void setup( )
{
  size( 400, 400 );
  strokeWeight( 3 );
  noFill( );
  rectMode( CENTER );
}

void draw( )
{
   background( 0 );
   drawAllDesigns( spacingList1, colorList );
   //drawAllDesigns( spacingList2, colorList );
   //drawAllDesigns( spacingList3, colorList );
   noLoop( );
}
```

- Copy your `design( )` function from question 1 into the program.
- Copy your `drawAllDesigns( )` from question 2 into the program.
- Modify the signature of your `drawAllDesigns( )` function to accept arguments so it can traverse different arrays to draw the design.
- You may not write three different functions do to this. One definition of `drawAllDesigns( )` must be able to traverse any one of the three `spacingList` arrays.

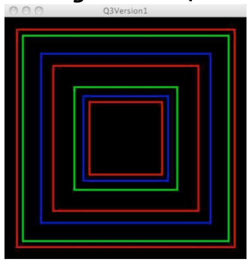
Expected outputs are on the following page:

The following versions of the `draw( )` function produce the following outputs:

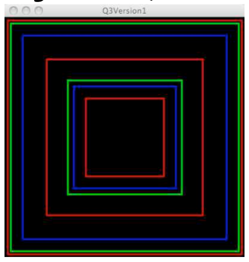
```
drawAllDesigns( spacingList1, colorList );
//drawAllDesigns( spacingList2, colorList );
//drawAllDesigns( spacingList3, colorList );
```



```
//drawAllDesigns( spacingList1, colorList );
drawAllDesigns( spacingList2, colorList );
//drawAllDesigns( spacingList3, colorList );
```



```
//drawAllDesigns( spacingList1, colorList );
//drawAllDesigns( spacingList2, colorList );
drawAllDesigns( spacingList3, colorList );
```



// Question 4 - 15 points

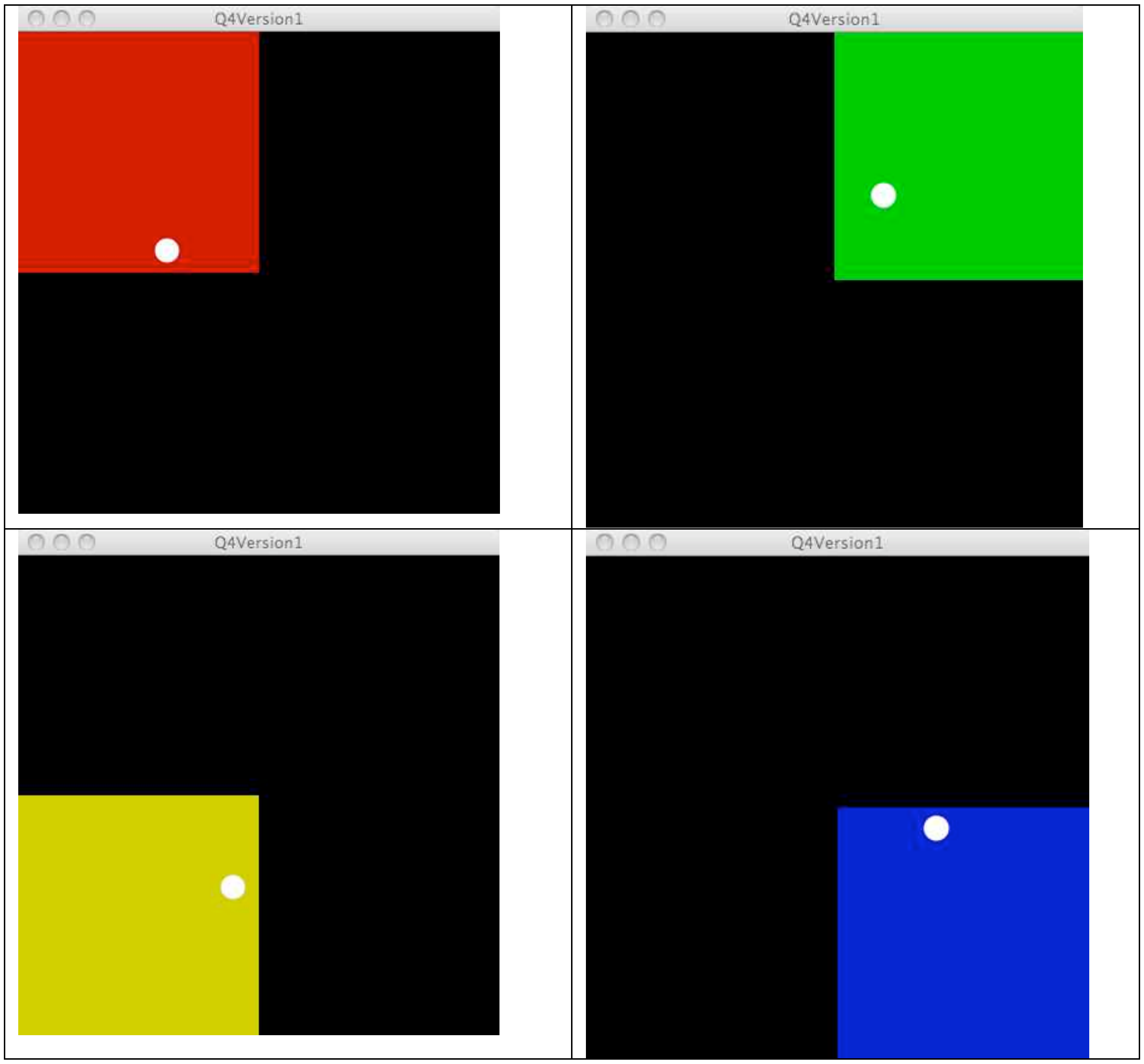
// Copy the following code into a new program:

```
int x, y, length;
int rotatingX, rotatingY;
int diam;
color ul, ur, ll, lr;

void setup( )
{
   size( 400, 400 );
   x = width/2;
   y = height/2;
   length = width/5;
   diam = 33;
   smooth( );
        // red for upper left quadrant
   ul = color( 200, 0, 0 );
        // green for upper right quadrant
   ur = color( 0, 200, 0 );
        // yellow for lower left quadrant
   ll = color( 200, 200, 0);
        // blue for lower right quadrant
   lr = color( 0, 0, 200);
}

void draw( )
{
   background( 0 );
   rotatingX =
       x + int( cos(radians(frameCount*2) ) * length );
   rotatingY =
       y + int( sin(radians(frameCount*2) ) * length );
   coloredBackground( rotatingX, rotatingY);
   stroke( 255 );
   line( x, y, rotatingX, rotatingY);
   fill( 255 );
   noStroke( );
   ellipse( width/2, height/2, 10, 10 );
   ellipse( rotatingX, rotatingY, 20, 20 );
}
void coloredBackground( int rotx, int roty )
{

}
```

Finish the definition of `coloredBackground( int rotx, int roty )`.

The arguments are the location of the rotating circle. The function must use the argument values to determine which quadrant the circle is in and fill that quadrant with the corresponding quadrant color.

Sample output – the four images below are taken at different times during the run of the program:



Old Exam 2 Version 2: Question 1 - 10 points

Copy the following code into a Processing program:

```
color c1, c2;

void setup( )
{
  size( 400, 400 );
  c1 = color( 200, 0, 0 );  // red
  c2 = color( 0, 0, 200 );  // blue
  strokeWeight( 3 );
}

void draw( )
{
   fill( c2 );
   triangle( 0, 0, width, height, 0, height);
   design( 100, c1 ); // 100 pixel separation & red lines
   design( 80, c2 );  //  80 pixel separation & blue lines
   noLoop( );
}
```

Define the function `design( int,  color );`

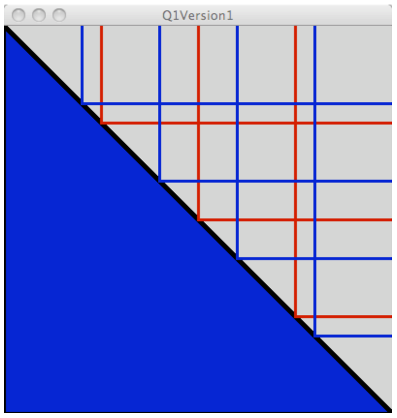
where the arguments are:

argument #1 is the space between the lines in the design

argument #2 is the color of the lines in the design

The window is always square! One half (the lower left area) of the window is a filled triangle – this code is already written. The design function you must define is a series of vertical and horizontal lines. The vertical lines extend from a diagonal line up to the top edge. The horizontal lines extend from the same diagonal line across to the right edge of the window. The first argument (the int) is the space between the lines. The second argument is the color of the lines. The stroke weight is set to the proper value in the setup( ) function.

Expected output:



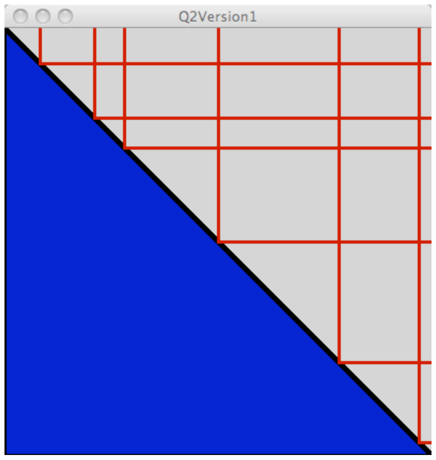
Question 2 - 10 points

Copy the following code into a Processing program:

```
int [ ] spaceValues = { 33, 84,  112,  200, 313, 388 };
color c1, c2;

void setup( )
{
  size( 400, 400 );
  c1 = color( 200, 0, 0 );  // red
  c2 = color( 0, 0, 200 );  // blue
  strokeWeight( 3 );
}

void draw( )
{
   fill( c2 );
   triangle( 0, 0, width, height, 0, height);
   design( c1 );// red lines
   noLoop( );
}
```

Recode your design( ) function so that it traverses the array and draws the lines at the locations specified in the array.
The argument is the color of the lines

Expected output:



If you could not get design code in problem 1 to work, you can code the design function for this problem by just drawing vertical lines at the proper spacing.

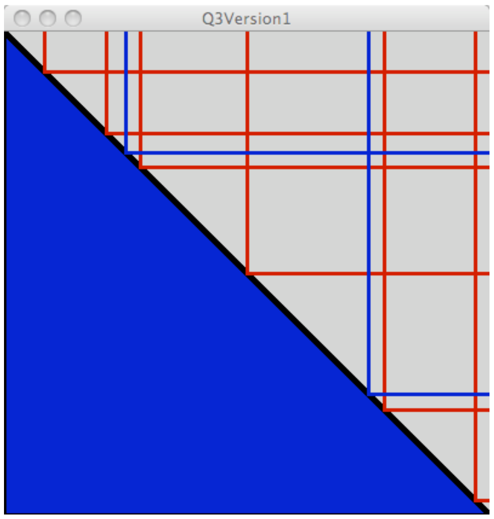
Question 3 - 15 points

Copy the following code into a Processing program:

```
int [ ] spacing1 = {   33,   84,  112,  200, 313, 388 };
int [ ] spacing2 = {  100,  300 };
color c1, c2;

void setup( )
{
  size( 400, 400 );
  c1 = color( 200, 0, 0 );  // red
  c2 = color( 0, 0, 200 );  // blue
  strokeWeight( 3 );
}

void draw( )
{
   fill( c2 );
   triangle( 0, 0, width, height, 0, height);
   design( spacing1, c1 );  // red lines
   design( spacing2, c2 );  // blue lines
   noLoop( );
}
```

Recode your design( ) function to take an array of it as it first argument. The second argument is the color of the lines.

Expected output:



If you could not get design code in problem 1 to work, you can code the design function for this problem by just drawing vertical lines at the proper spacing.

Question 4 - 15 points

Copy the following code into a Processing program:

```
int x, y, length;
int rotatingX, rotatingY;
int diam;
color ul, ur, ll, lr;

void setup( )
{
   size( 400, 400 );
   x = width/2;
   y = height/2;
   length = width/5;
   diam = 33;

   stroke( 255 );
   strokeWeight( 3 );
   smooth( );
        // red for upper left quadrant
   ul = color( 200, 0, 0 );
        // green for upper right quadrant
   ur = color( 0, 200, 0 );
        // yellow for lower left quadrant
   ll = color( 200, 200, 0);
        // blue for lower right quadrant
   lr = color( 0, 0, 200);
}

void draw( )
{
   background( 0 );

   rotatingX =
       x + int( cos(radians(frameCount*2) ) * length );

   rotatingY =
       y + int( sin(radians(frameCount*2) ) * length );

   line( x, y, rotatingX, rotatingY);

   coloredEllipse( rotatingX, rotatingY );
}
```

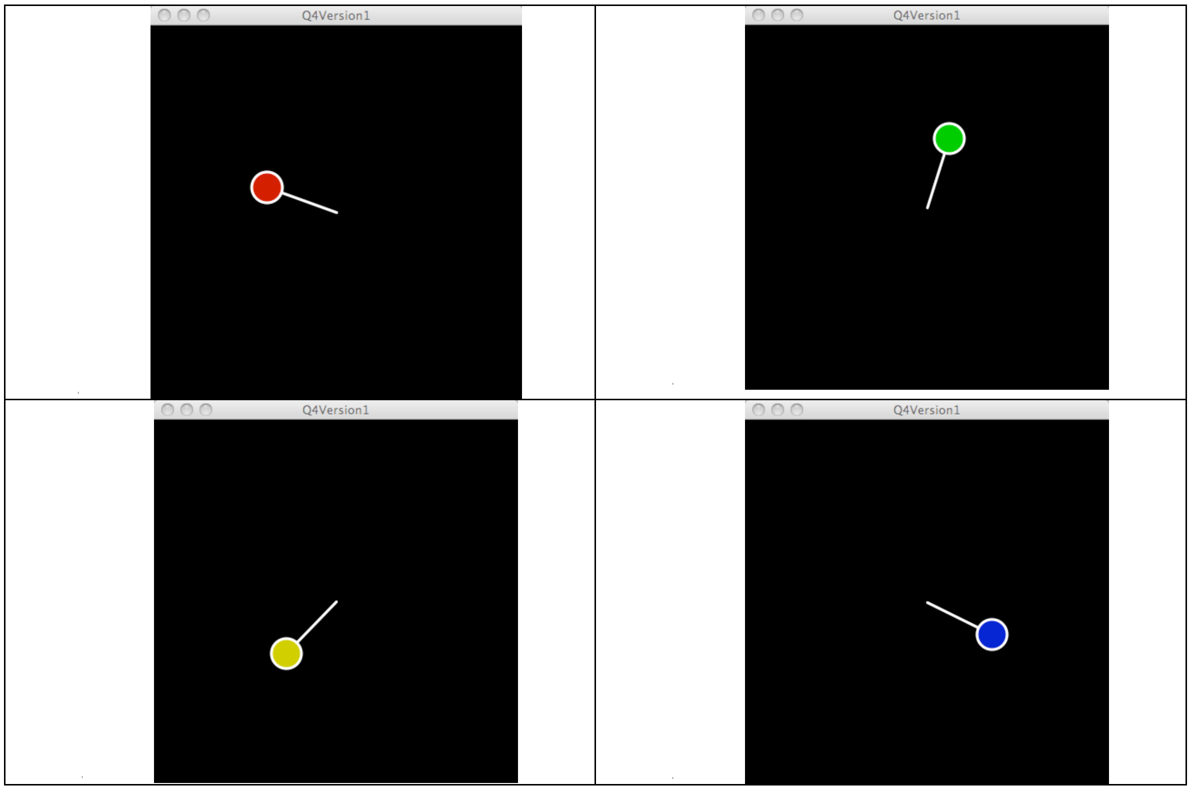
This code draws a line that rotates around the center of the window. How the rotation works is not important to this question. Your task is to define the function:

```
        coloredEllipse( rotatingX, rotatingY );
```

that draws a filled circle at the end of the rotating line at the location specified by the arguments. The diameter of the circle is specified in the setup( ) function as the `diam`. Here are the coloring rules for the circles:

- upper left quadrant circle is red
- upper right quadrant is green
- lower left quadrant is yellow
- lower right quadrant is blue

Here is the expected output as the line rotates and the color circles are drawn during the rotation:



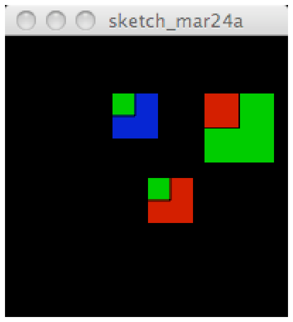
Old Exam 2 Version 3: Question 1 - 10 points

Copy the following code into a processing program:

```
color c1, c2, c3;
void setup ( )
{
  size( 200, 200 );
  noFill( );
  smooth( );


  c1 = color( 200, 0, 0 );  // red
  c2 = color( 0, 200, 0 );  // green
  c3 = color( 0, 0, 200 );  // blue
}

void draw ( )
{
  background ( 0 );
  figure(  100, 100, 33, c1, c2 );
  figure(  140,  40, 50, c2, c1 );
  figure(   75,  40, 33, c3, c2 );
}
```

Define the function `figure( float, float, float, color, color)`
where the arguments are:
argument #1 is the x coordinate of both rectangles
argument #2 is the y coordinate of both rectangles
argument #3 is the edge length of the larger rectangle
the edge length of the smaller rectangle is one half of argument #3
argument #4 is the color of the larger rectangle
argument #5 is the color of the smaller rectangle
Do not worry about the stroke color, the stroke weight, or the background color.

Expected output:



Question 2 - 10 points

Make a copy of the program you wrote for question 1 and use it to answer this question.
- Copy the following array declarations into your code as global variables:

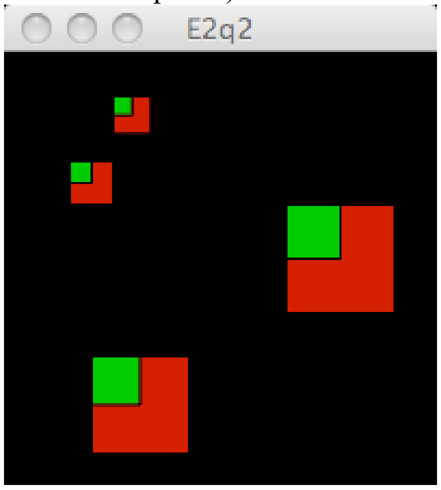
```
float  [ ] xCoordinates = {  40, 130, 30, 50 };
float  [ ] yCoordinates = { 140,  70, 50, 20 };
```

Modify your `draw( )` function to look like this:

```
void draw( )
{
  background ( 0 );
  drawAllFigures( );
}
```

- Define the function, `drawAllFigures( )` so that it traverses the arrays and draws figures using the data in the arrays.
- Your `drawAllFigures( )` must use the `figure( )` function you defined in question 1. The edge size of the rectangles (argument #3) must be 25% of the sum of the x and y coordinates of the figure. For the zeroth figure, the edge length would be 45 pixels.
  If you were not able to get question one working, you can code the functions calls to draw the figure inside the loop that is traversing the arrays for partial credit.
- The colors are up to you.
- You MAY NOT hardwire the loop to work with only four-element arrays.
- Your code must work with parallel arrays of int of any size.

Expected output (colors seen are not required):



Question 3 - 15 points

Make a copy of the program you wrote for question 2 and use it to answer this question.
- Copy the following array declarations into your code as global variables:

```
float  [ ] x1Cords = { 40, 90, 130, 150 };
float  [ ] y1Cords = { 50, 40,  30,  60 };

float  [ ] x2Cords = {  20,  50, 100 };
float  [ ] y2Cords = {  90, 110, 100 };

float  [ ] x3Cords = {  80, 160, 110, 20, 80 };
float  [ ] y3Cords = { 130,  50,  80, 90, 77 };
```

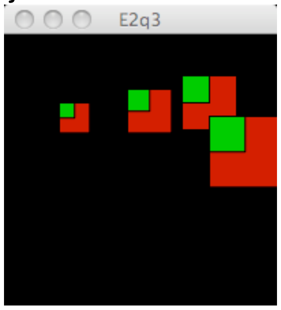
- Modify the `drawAllFigures( )` function wrote in question #2 so that it can use any two parallel arrays as arguments to draw the group of figures.
- The pseudo-signature of the modified function must be this:
  `drawAllFigures(array-x-coordinates, array-y-coordinates, color, color)`
- You MAY NOT define three different functions to answer this question. You must use the same function (with different arguments) three times as shown below.
  Modify your `draw( )` function to look something like this:

```
void draw( )
{
  background ( 0 );
  drawAllFigures( x1Cords, y1Cords,  c1, c2);
  //drawAllFigures( x2Cords, y2Cords,  c2, c3);
  //drawAllFigures( x3Cords, y3Cords,  c3, c1);
}
```

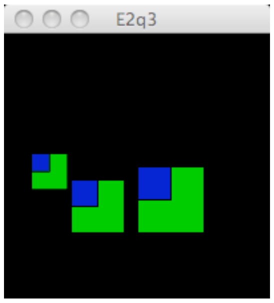
Expected outputs are on the following page:

The following versions of the `draw( )` function produce the following outputs:

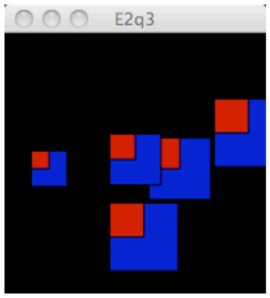
```
void draw( )
{
  background ( 0 );
  drawAllFigures( x1Cords, y1Cords,  c1, c2);
  //drawAllFigures( x2Cords, y2Cords,  c2, c3);
  //drawAllFigures( x3Cords, y3Cords,  c3, c1);
}
```



```
void draw( )
{
  background ( 0 );
  //drawAllFigures( x1Cords, y1Cords,  c1, c2);
  drawAllFigures( x2Cords, y2Cords,  c2, c3);
  //drawAllFigures( x3Cords, y3Cords,  c3, c1);
}
```



```
void draw( )
{
  background ( 0 );
  //drawAllFigures( x1Cords, y1Cords,  c1, c2);
  //drawAllFigures( x2Cords, y2Cords,  c2, c3);
  drawAllFigures( x3Cords, y3Cords,  c3, c1);
}
```



// **Question 4 - 15 points**

// Copy the following code into a new program and finish the `drawFigures( )`
// function so the figures change color when they move into different quadrants:

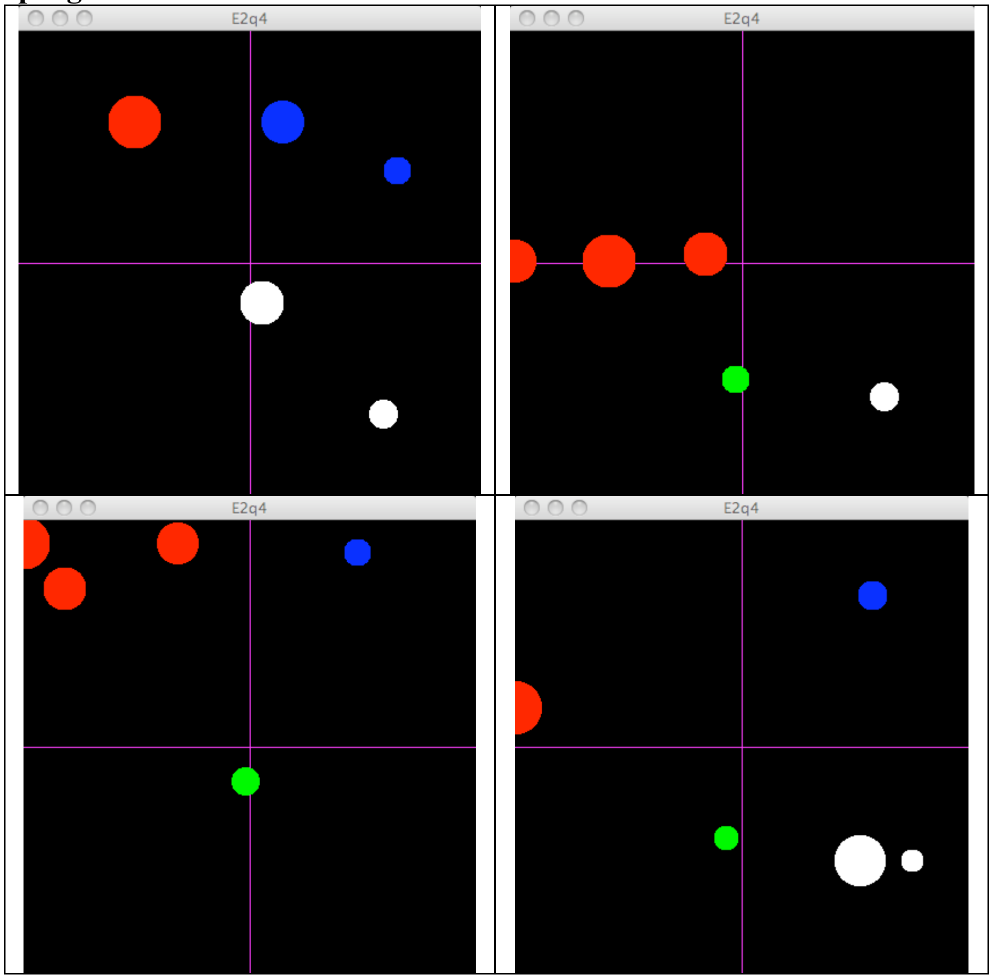
// upper left quadrant is red	--	fill(255, 0, 0);
// upper right quadrant is blue	--	fill(0, 0, 255);
// lower left quadrant is green	--	fill(0, 255, 0);
// lower right quadrant is white	--	fill(255);

// The magenta lines show the quadrant boundaries of the window

```
final int MAX = 5;
int [ ]  x, y, dx, dy, diam;

void setup ( )
{
  size( 400, 400 );
  x = new int[ MAX ];
  y = new int[ MAX ];
  dx = new int[ MAX ];
  dy = new int[ MAX ];
  diam = new int[ MAX ];

  initializeIntArray( x, 0, width );
  initializeIntArray( y, 0, 1 );
  initializeIntArray( dx, 1, 8 );
  initializeIntArray( dy, 1, 8 );
  initializeIntArray( diam,15, 50 );
}

void initializeIntArray( int [ ] array, int small, int big )
{
  for ( int i = 0; i < array.length; i++)
  {
     array[i] = int(random( small, big ) );
  }
}

void draw ( )
{
    prepareWindow( );
    moveFigures( );
    drawFigures( );
}

void prepareWindow( )
{
   background( 0 );
   stroke( 255, 0, 255);
   line( width/2, 0, width/2, height );
   line( 0, height/2, width, height/2);
   noStroke( );
}
```

```
void moveFigures( )
{
   for ( int i = 0; i < x.length; i++)
   {
      x[i] += dx[i];
      if( x[i] > width)  x[i] = 0;
      y [i] += dy[i];
      if(y[i] > height) y[i] = 0;
   }
}

void drawFigures( )
{

}
```

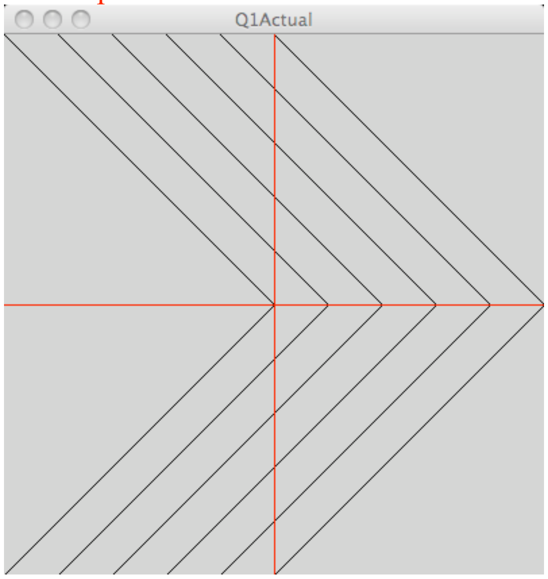
Sample output – the four images below are taken at different times during the run of the program:



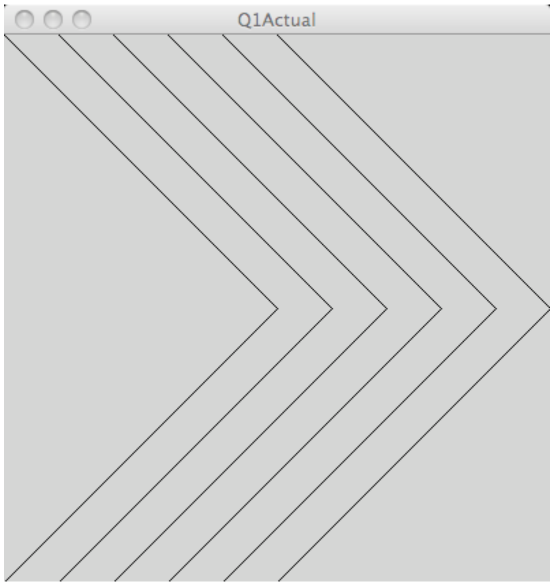
Old Exam 2 Version 4:
Question 1 5 points Back to Homework #1

Write a program without setup( ) and draw( ) functions (as you wrote the first few homeworks) and with no user input to draw the figure shown below. The window size is always square and never changes. Draw it any way you wish – nothing is illegal as long as it works.

Do not draw the red lines – they are shown in the figure below to allow you to determine the relative position of the end points of the lines..



Your image should look like this:

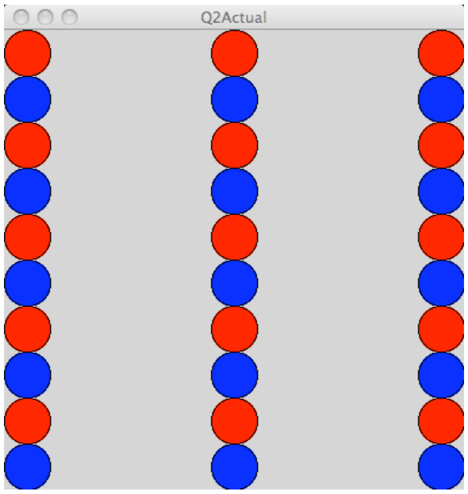


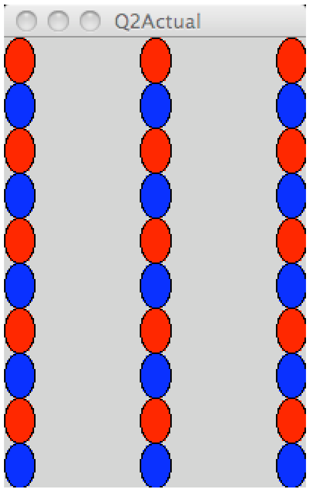
Question 2 10 points Control

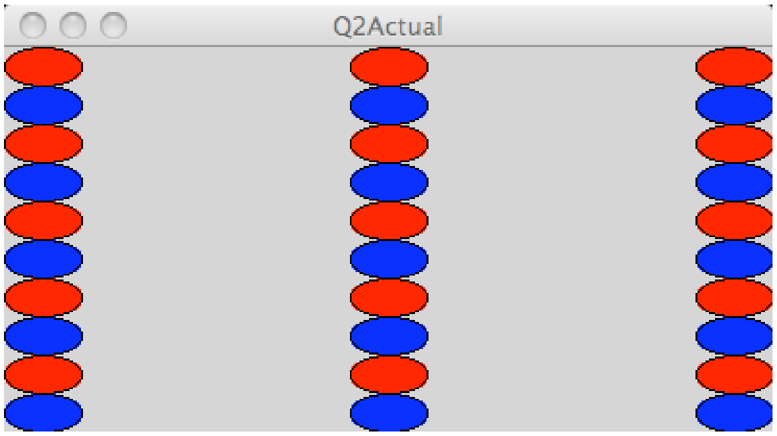
Write a program using setup() and draw( ) functions. The program must do the following:

- draw three vertical columns of ellipses.
- there must ten ellipses in each column.
- the color of the ellipses must alternate between red and blue.
- the top ellipse must be red.
- the ellipses must be completely on screen.
- the ellipses must adjust to different size windows.

Below are three runs with different window sizes:





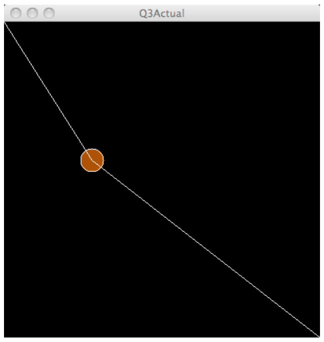


HINT #1: This reeks of the need for a loop or loops and an if/else – 30 calls to ellipse is a VERY BAD idea.
HINT #2: An int value divided by 2 using the % operator (which gives you the remainder of the division) evaluates to 0 for even values and 1 for odd values.

Question 3 15 points Defining Functions

Write a program using setup() and draw( ) functions. The program must do the following:

- define a function that draws this figure:



- The figure is a filled, colored, circle connected to the upper left corner and the lower right corner by two lines.
- The color of the two lines is up to you.

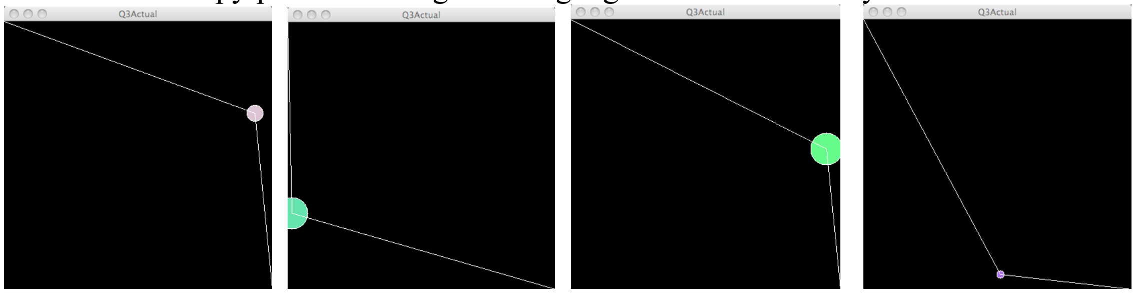
- Values for the diameter, x, y, and color can be global variables.
- The diameter of the circle is random but reasonable.
- The color of the circle is random.
- The x position is random and can show partial circles on the edges.

- Define a function that moves circle from the top to the bottom with a random deltaY that is small in value.
- When the circle reaches the bottom of the window, its x, diameter, deltaY, and color values must be randomly reset and the circle repositioned to the top of the window.

- The background color is up to you.
- Call the functions to move and draw the figure from the draw( ) function.

- The use of arguments for the function that draws the figure is optional here because there is only one figure and you are using global variables. It is up to you.

Here are four different screen shots. The broken white lines connecting the circles are an artifact of the copy/paste and image resizing algorithms and used by Word.



HINT #3:

```
    someVariable = color( random(255), random(255), random(255) );
```

will set a color variable to a random color.

Question 4 - 20 points Arrays

Copy your question 3 folder and rename it question4. Rename the .pde file if you have to.

Modify your code to move five different figures in the same way the single figure moved in question 3. Edit your functions to move and draw the figures to work the arrays or the global variables as needed.

You have two options for doing this: One requires much less code but involves parallel arrays. The other involves lots of global variables (one set for each figure) and a great deal of code. You are strongly urged to use arrays.

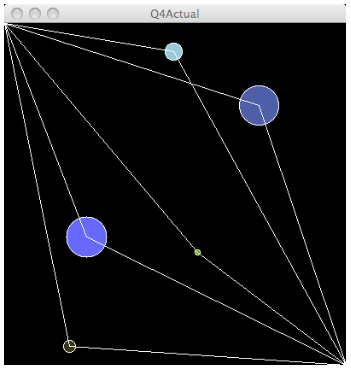
- The array option requires global arrays for the x, y, deltaY, and color* values. You may use initializer lists or new the arrays and call initialize functions – this is up to you.

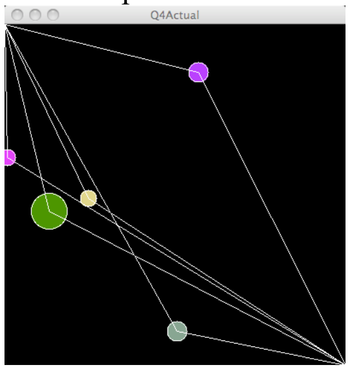
- The individual global variables option requires 20 global variables – five for each figure.

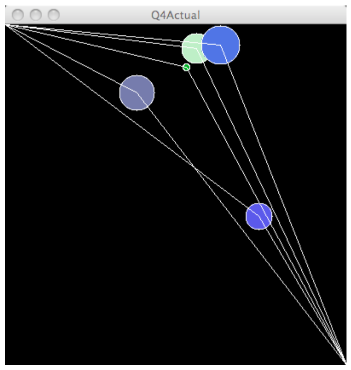
If you could not get the function that draws the figure to work properly in question 3, you can still do this problem with a simple rectangle, but there will be a deduction.

If you could not get the function that moves the figure to work properly in question 3, you can code it in draw, but there will be a deduction.

Below is three screen shots of the sample solution.







*HINT #5: If you cannot get the array of color working, just make all of the figures the same color. This would be a small deduction but you would still get most of the points.