# Old 257 Exam Ones for Practice

**Exam 1 will be taken on Thursday February 13 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. You may not stay late because there is a class following ours.**

**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 1-5. Homework 5's 3-D stuff is NOT part of the exam.**

**You will have access to the Processing API. You MAY NOT access any notes or 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.**

**The following contains four different exam ones from the four previous terms.**

**Remember that the order of topics and homeworks can vary from semester so some of the questions on the old exams may not be pertinent to the material covered in this term's homeworks 1-5.**

# 257 Programming in the Arts with Processing
## Fall 2013 Exam 1 Sample for Practice
## Question 1 5 points Locating Positions with Expressions

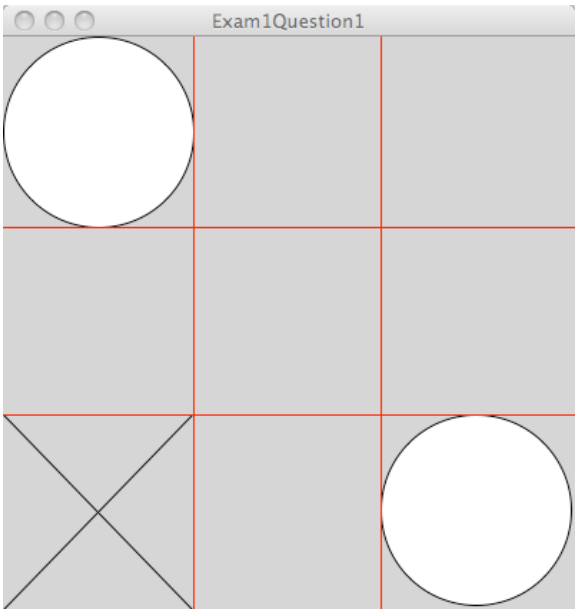
Write a program without **`setup()`** and **`draw()`** functions ( as you wrote the first two homeworks) and with no user input to draw the figure shown below.

The figure must adjust to fit into different window sizes without any recoding other than changing the window size. Because of this requirement, the code must use the system variables width and height and expressions to locate the figures. It must not use magic numbers.

The color of the **`background`**, the **`fill`**, and the **`stroke`** are the default colors – do not worry about them. The **`strokeWeight`** has a default of 1. Do not alter these values.

**DO NOT** draw the red lines shown below. The are there to give you some guidance for locating the elements of the figure.

Your figure should look like this (without the red lines):



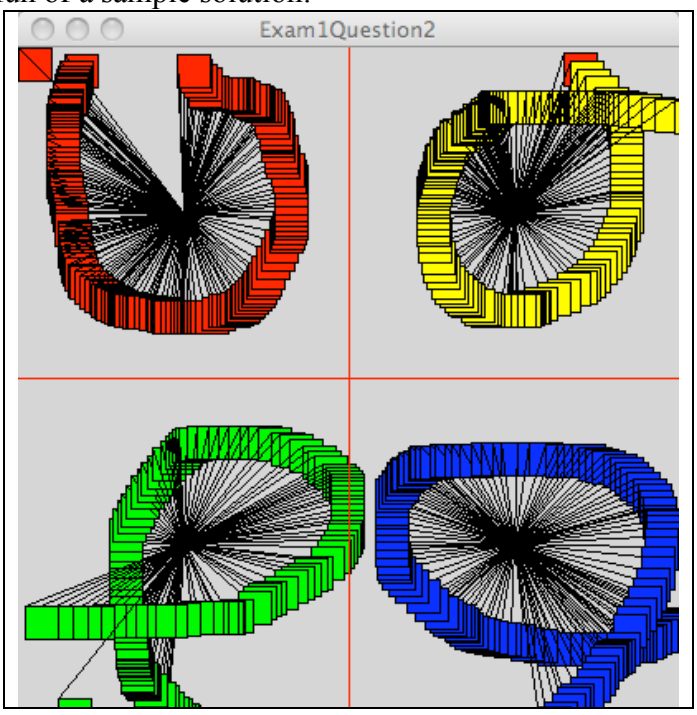
## Question 2 10 points Choosing What to Do

Write a program using `setup()` and `draw()` functions.

The program must do the following:

- Draw a small sized (up to you) filled rectangle in every frame that is at the location of the mouse. The rectMode() can be the default CORNER mode or you can change it to CENTER. It does not matter.
- The color of the rectangle must be based on the quadrant in which the rectangle is drawn. The four quadrants are the upper left, upper right, lower left, and lower right. The colors are up to you. We suggest red (255, 0, 0 ), green ( 0, 255, 255), blue ( 0, 0, 255), and yellow( 255, 255, 0 ).
- Draw one line that extends from some point in the rectangle to the center of quadrant in which the rectangle is drawn. Where the line makes contact with the rectangle is up to you as long as it contacts the rectangle.
- Do not draw the red lines – they are there to show you the quadrant boundaries.

Here is a sample fun of a sample solution:




**Notes:**

- There will be a rectangle in the upper left corner until you move the mouse into the frame.

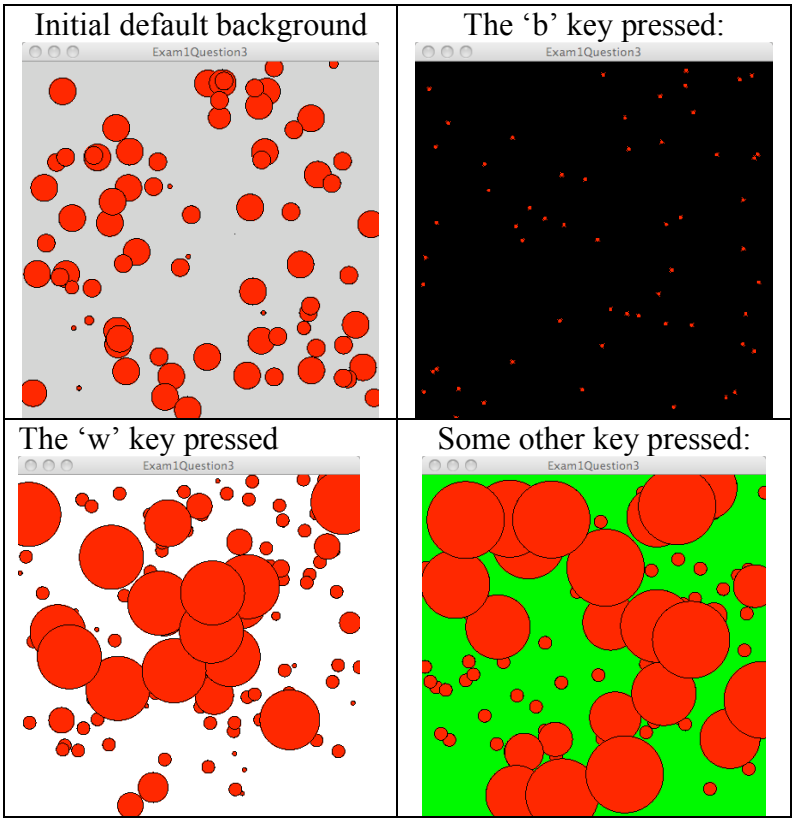
- The first rectangle drawn in a new quadrant may be the wrong color and may have a line from the wrong quadrant– this is fine.

# Question 3 15 points Responding to User Input

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

- Declare global variables to store the position, and size values for a circle or square.
- Initialize the position variables to random values that are in the window. Use the **`random( )`** function to do this.
- Initialize the size variable to some small reasonable value such as 50.
- Initialize the **`fill`** to any color of your choice.
- Leave the background color its default value.
- Each frame must draw the circle or square and then move it to a new random position.
- If the user **DRAGS** (i.e. the button is down) the mouse, increase the size of the circle or rectangle.
- If the user **MOVES** (i.e. the button is up) the mouse, decrease the size of the circle or rectangle. **Note that if you decrease the circle or rectangle to zero, further decreases actually make the circle or rectangles bigger. This is ok – it is not a bug in your code or Processing.**
- If the user presses a key, respond as follows:
  - the '**b**' key – call background( 0 );
  - the '**w**' key – call background( 255 );
  - any other key – call background with any color of your choosing.

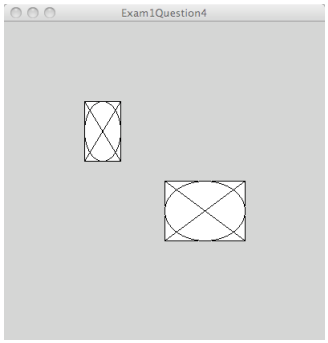
Here are four different screens after the user has moved or dragged the mouse or pressed a key:



# Question 4 - 20 points Defining a Function with Arguments

Write a program using **`setup()`** and **`draw()`** functions.
You **MAY NOT USE** GLOBAL VARIABLES*.
The program must do the following:

- define a function named **`figure()`** that draws figures similar to the two shown below:

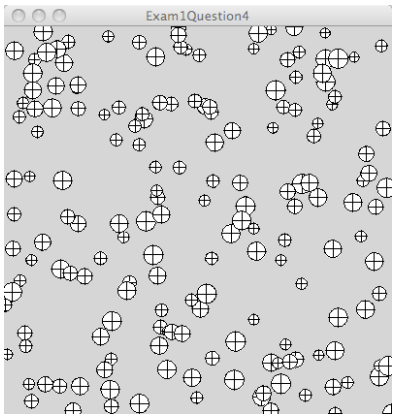



The figure is made up of a rectangle, an ellipse, and two lines.

- The function must have four arguments:
  - two for the **`(x, y)`** coordinates for the anchor point – the position of the anchor point for the figure is up to you – it is not specified by the exam.
  - two for the size of the figure – the width and height.
- The **`stroke`**, **`strokeWeight`** and **`fill`** are the default values.
- One call of the **`figure( )`** function results in ONLY ONE figure being drawn.
- The function **`draw()`** calls the **`figure()`** function using random numbers to position the figure and reasonable random numbers for the width and height of the figure. Here is what your **`draw()`** function must look like:

```
void draw ( )
{
  figure(random(0, width), random(0, height),
         random(10, 20)   , random 10, 20);
}
```

Here is a run of a sample solution:




* If you cannot write the function **`figure( )`** with arguments as required, you may write it using global variables. This will be a point deduction but it will be **MUCH** better than taking a zero for the question.

# 257 Programming in the Arts with Processing
# Spring 2013 Exam 1 Sample for Practice

## Question 1 5 points Locating Positions with Expressions

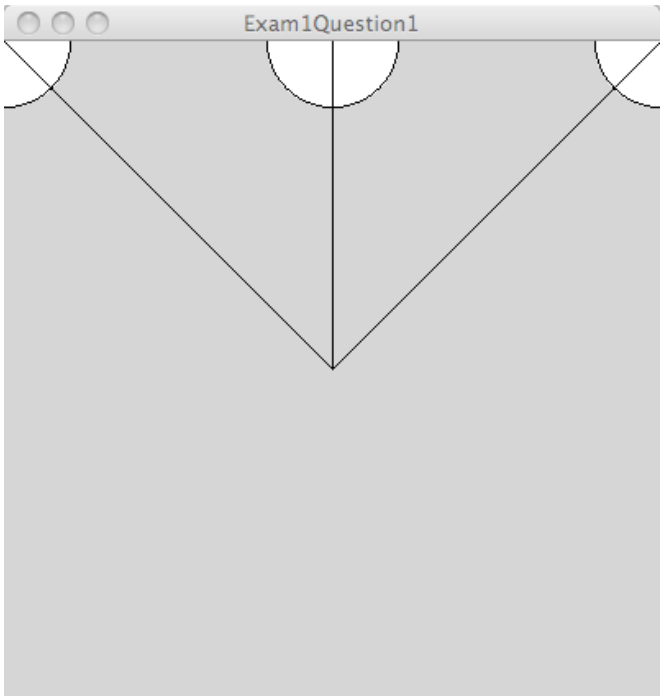
Write a program without **`setup()`** and **`draw()`** functions ( as you wrote the first two homeworks) and with no user input to draw the figure shown below.

The code must use variables and expressions to locate the figures. It must not use magic numbers.

The figure must adjust to fit into different window sizes without any recoding other than changing the window size.

The color of the **`background`**, the **`fill`**, and the **`stroke`** are the default colors – do not worry about them. The **`strokeWeight`** has a default of 1. Do not alter these values.

Your figure should look like this:



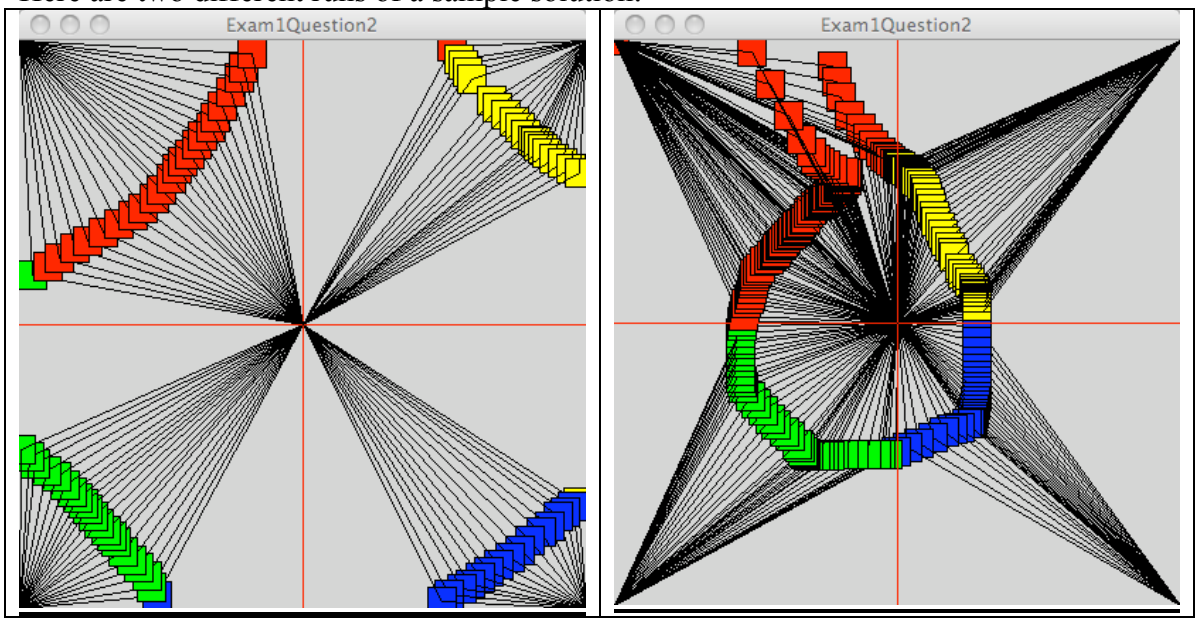
## Question 2 10 points Choosing What to Do

Write a program using `setup()` and `draw()` functions.

The program must do the following:

- Draw a small sized filled rectangle in every frame that is at the location of the mouse. The rectMode() can be the default CORNER mode or you can change it to CENTER. The code used to make the figures had the rectMode( ) set to CENTER. It does not matter.
- The color of the rectangle must be based on the quadrant in which the rectangle is drawn. The four quadrants are the upper left, upper right, lower left, and lower right. The colors are up to you. We suggest red (255, 0, 0 ), green ( 0, 255, 255), blue ( 0, 0, 255), and yellow( 255, 255, 0 ).
- Draw two lines must to some point within the rect.
  - one line must extend from the center of the window
  - one line must extend from the corner closest to the rectangle
- Do not draw the red lines – they are there to show you the quadrant boundaries.

Here are two different runs of a sample solution:




Notes:

- There will be a rectangle in the upper left corner until you move the mouse into the frame.

- The first rectangle drawn in a new quadrant may be the wrong color – this is fine.

## Question 3 15 points User Input
## This question is not appropriate this term because we have not done wrapping motion

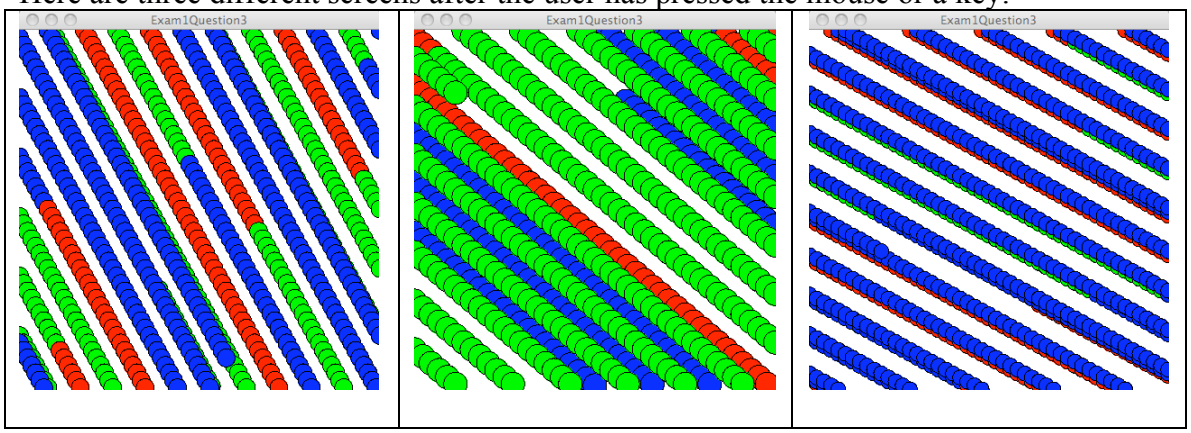
Write a program using **`setup()`** and **`draw()`** functions.

The program must do the following:

- Declare variables to store the position, size, and deltaX and deltaY values for a circle or square. There must be two delta variables.
- Initialize the variables to random values as follows:
  - x and y to any position in the frame
  - size variable/s between 5 and 30
  - delta variables between 3 and 10

  Use the **`random( )`** function to do this.
- Initialize the **`fill`** to white ( 255 ).
- Each frame must draw the circle or square and then move it to a new position that is computed by adding the delta values to the x and y variables.
- If the circle or square moves off the right side or the bottom side, it must be wrapped back to the left side or the top side.
- If the user presses a key, change the fill as follows:
  - the '**r**' key changes the fill to red (255, 0, 0 )
  - the '**g**' key changes the fill to green ( 0, 255, 0 )
  - the '**b**' key changes the fill to blue ( 0, 0, 255 )
  - any other key press, change the fill to white ( 255 ).
- If the user presses the mouse, the variables used to draw and move the square or circle are assigned new random values as listed above.
  DO NOT CALL **`setup()`** TO DO THIS!

Here are three different screens after the user has pressed the mouse or a key:



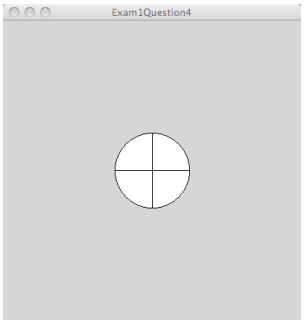
# Question 4 - 20 points Defining a Function with Arguments

Write a program using **`setup()`** and **`draw()`** functions.

You **MAY NOT USE** GLOBAL VARIABLES*.

The program must do the following:

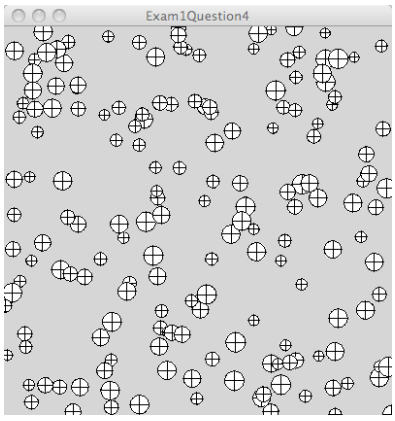
- define a function named **`figure()`** that draws the figure shown below




- The function must have three arguments:
  - two for the **`(x, y)`** coordinates
  - one for the size of the figure.
- The figure is composed of a circle and two lines.
- The anchor point location of the **`(x, y)`** coordinates is up to you.
- The **`stroke`**, **`strokeWeight`** and **`fill`** are the default values.
- One call of the **`figure( )`** function results in ONLY ONE figure being drawn.
- The function **`draw()`** calls the **`figure()`** function using random numbers to position the figure and a reasonable random number for the size of the figure. Here is what your **`draw()`** function must look like:

```
void draw ( )
{
  figure(random(0, width), random(0, height), random(10, 20));
}
```

Here is a run of a sample solution:

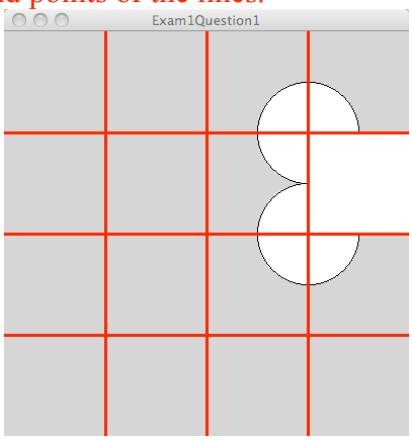



* If you cannot write the function **`figure( )`** with arguments as required, you may write it using global variables. This will be a point deduction but it will be **MUCH** better than taking a zero for the question.

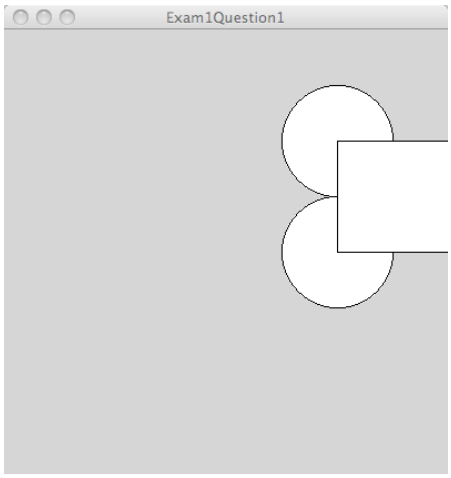
# 257 Programming in the Arts with Processing
# Fall 2012 Exam 1 Sample for Practice

## Question 1 5 points Locating Positions with Expressions

Write a program without **`setup()`** and **`draw()`** functions ( as you wrote the first homeworks) and with no user input to draw the figure shown below. The code must use variables and expressions to locate the figure. It must not use magic numbers. The figure must adjust to fit into different window sizes without any recoding other than changing the window size. The color of the **`background`**, the **`fill`**, and the **`stroke`** are the default colors – do not worry about them. The **`strokeWeight`** has a default of 1. Some lines look wider than others in the figures below because of a bug in Processing. 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 figure should look like this:

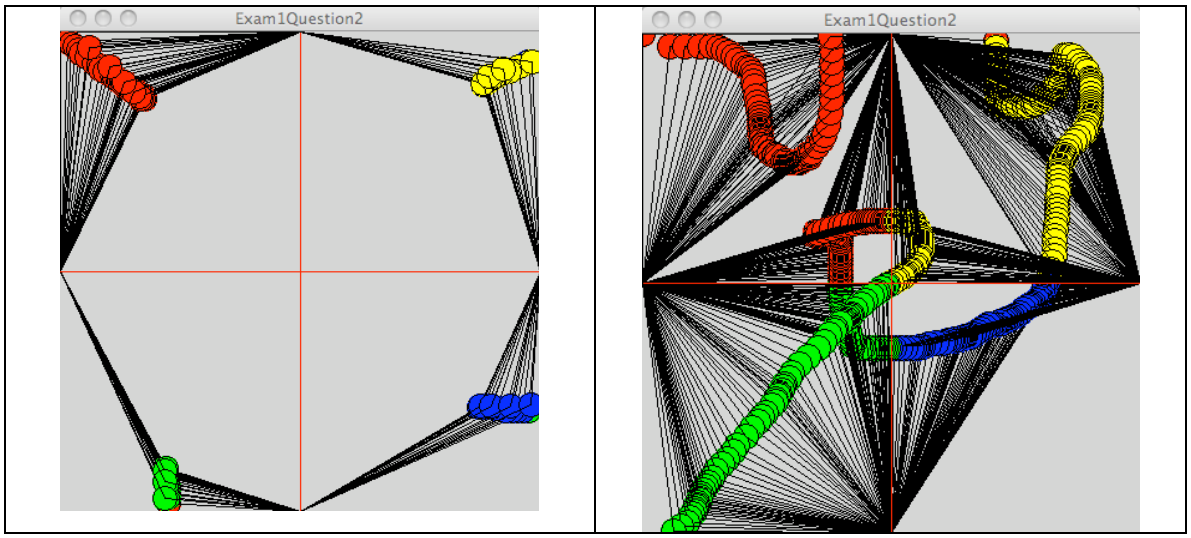


## Question 2 10 points Choosing What to Do

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

- Draw a small size filled circle in every frame at the location of the mouse.
- The color of the circle must be based on the quadrant in which the circle is drawn. The four quadrants are the upper left, upper right, lower left, lower right. The colors are up to you. We suggest red (255, 0, 0 ), green ( 0, 255, 255 ), blue ( 0, 0, 255), and yellow( 255, 255, 0 ).
- Draw two lines connecting the center of the circle to the center of the two closest window edges as follows:
  - If the mouse is in the upper left quadrant, the line extends from the middle of the top edge and the middle of the left edge.
  - If the mouse is in the upper right quadrant, the line extends from the middle of the top edge and the middle of the right edge.
  - If the mouse is in the lower left quadrant, the line extends from the middle of the bottom edge and the middle of the left edge.
  - If the mouse is in the lower right quadrant, the line extends from the middle of the bottom edge and the middle of the right edge.

Do not draw the red lines – they are there to show you the quadrant boundaries.



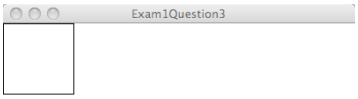
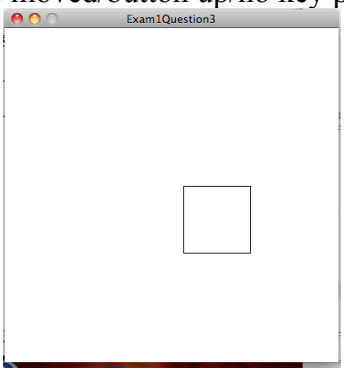
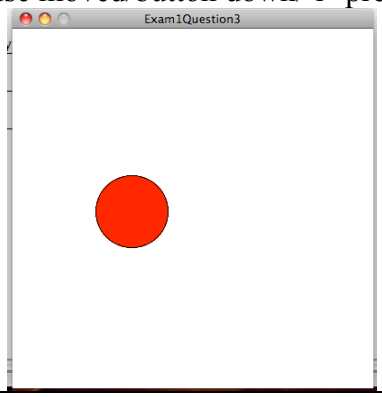
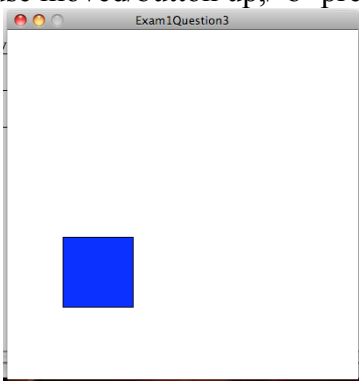

Note the first circle drawn in a new quadrant may be the wrong color – this is fine.

# Question 3 15 points User Input

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

- Declare variables to store the position and size of a circle or square.
- The initial position is up to you.
- The size of the circle and/or rectangle must be 20% of the width of the window.
- Set the **`fill`** to white ( 255 ).
- Use white for the background color ( 255 ).
- Each frame must have either a circle or a square as follows:
  -- the mouse button is pressed, a circle is drawn
  -- otherwise a square is drawn
- The location of the circle or square should be the location of the mouse. The mode does not matter.
- If the user presses the '**`r`**' key, change the fill to red (255, 0, 0 ).
- If the user presses the 'g' key, change the fill to green ( 0, 255, 0 ).
- If the user presses the 'b' key, change the fill to blue ( 0, 0, 255 ).
- Any other key press, change the fill to white ( 255 ).

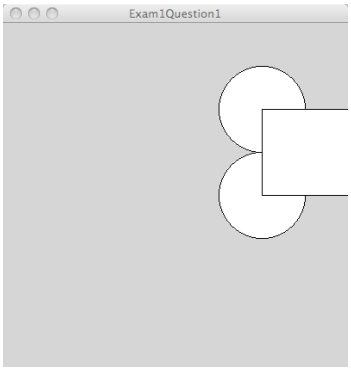
Here are four different screens displayed by the code:

<table>
<tr>
<td>Initial screen<br> 
</td>
<td>Mouse moved/button up/no key pressed<br> 
</td>
</tr>
<tr>
<td>Mouse moved/button down/'r' pressed<br> 
</td>
<td>Mouse moved/button up,/'b' pressed<br> 
</td>
</tr>
</table>

# Question 4 - 20 points Defining a Function with Arguments

Write a program using **`setup()`** and **`draw()`** functions. You **MAY NOT USE GLOBAL VARIABLES.** The program must do the following:

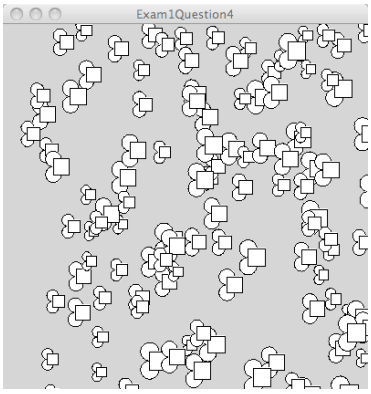
- define a function named **`figure()`** that draws the figure shown below




- The function must have three arguments:
  - two for the **`(x, y)`** coordinates
  - one for the size of the individual figures.
- The figure is composed of two circles and a square.
- The anchor point location of the **`(x, y)`** coordinates are up to you. It can be a corner or the middle of some component of the figure.
- The **`stroke`** and **`strokeWeight`** are the default values.
- One call of the figure( ) function results in ONLY ONE figure being drawn.
- The function **`draw()`** calls the **`figure()`** function using random numbers to position the figure and a reasonable random number for the size of the figure. Here is what your **`draw()`** function should look like:

```
void draw ( )
{
  figure(random(0, width), random(0, height), random(10, 20));
}
```

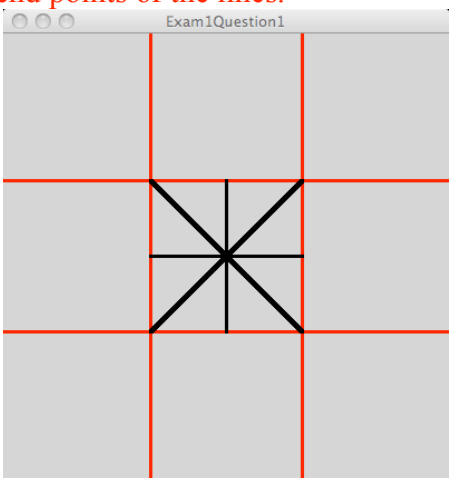
Here is a run of a sample solution:



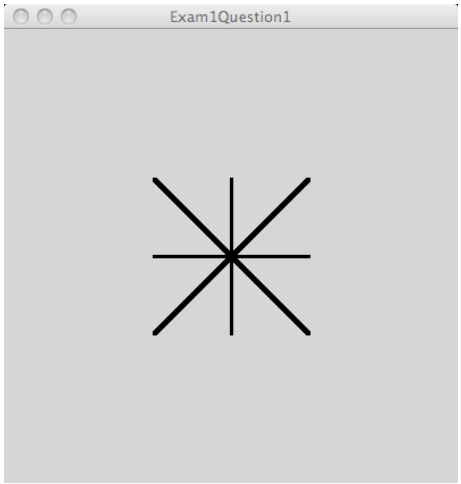
# 257 Programming in the Arts with Processing
# Spring 2012 Exam 1 Sample for Practice

## Question 1 5 points Locating Positions with Expressions

Write a program without **`setup()`** and **`draw()`** functions ( as you wrote the first homeworks) and with no user input to draw the figure shown below. The code must use variables and expressions to locate the figure. It must not use magic numbers. The figure must adjust to fit into different window sizes without any recoding other than changing the window size. The color of the **`background`**, the **`fill`**, and the **`stroke`** are the default colors – do not worry about them. The **`strokeWeight`** has a default of 1. The diagonal lines look wider in the figures below because of a bug in Processing. 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 figure should look like this:

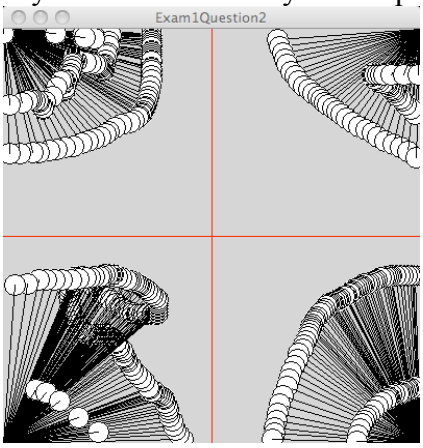


# Question 2 10 points Choosing What to Do

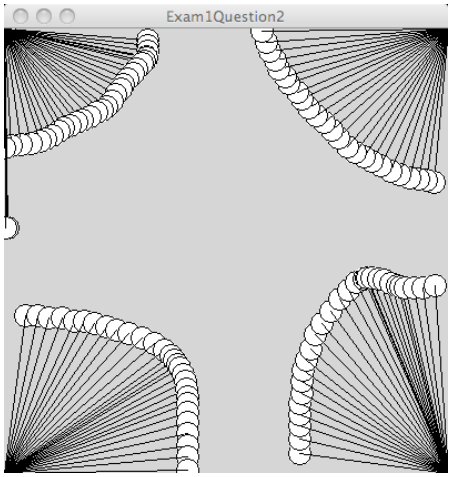
Write a program using **`setup()`** and **`draw()`** functions. The program must do the following:

- Leave the **`fill`**, **`stroke`** and **`strokeWeight`**s a the default values.
- Draw a small size circle in every frame at the location of the mouse.
- Draw a single connecting line from the center of the circle to the closest corner of the window.
- The line must be drawn from the upper left corner if the mouse is in the upper left quadrant of the window.
- The line must be from the upper right corner if the mouse is in the upper right quadrant.
- The line must be from the lower right corner if the mouse is in the lower right quadrant.
- The line must be from the lower left corner if the mouse is in the lower left quadrant.

Do not draw the red lines – they are there to show you the quadrant boundaries.




Your figure should look like this:

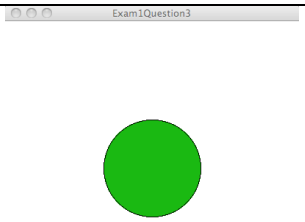


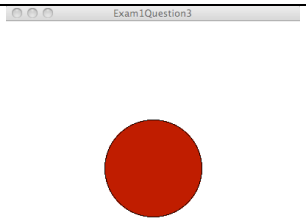
## Question 3 15 points User Input

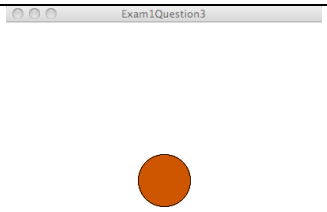
Write a program using **`setup()`** and **`draw()`** functions. The program must do the following:

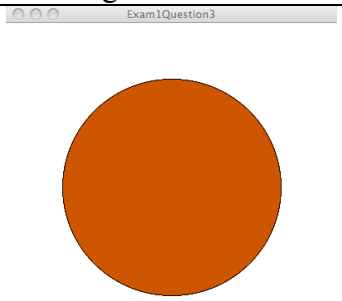
- Declare variables to store the position and size of a circle.
- Initialize the position variables to put the circle at the center of the window. Do this so that the circle will always be in the middle of the window regardless of the window's size.
- Initialize the size of the circle to be one third of the width of the window.
- Set the **`fill`** to a random color other than black/white/gray.
- Call the **`background()`** and **`ellipse()`** functions from **`draw()`**. Use white for the background color.
- If the user presses the mouse button, change the **`fill`** to a new random color.
- If the user presses the '**`g`**' key, double the size of the circle.
- If the user presses the '**`s`**' key, reduce the size of the circle by half. The circle may disappear if this is done many times – this is fine.

Here are four different screens displayed by the code:

<table>
<tr>
<td> 

The initial window</td>
<td> 

Color change after a mouse click</td>
</tr>
<tr>
<td> 

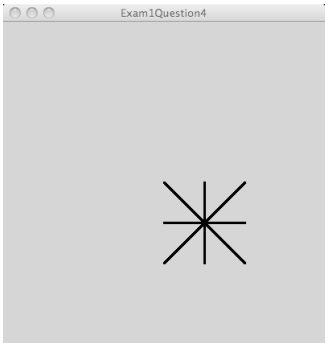
After one press of the 's' key – circle size is one half of the circles in the upper row</td>
<td> 

After two presses of the 'g' key when the circle was the size of the figure to the left</td>
</tr>
</table>

# Question 4 - 20 points Defining a Function with Arguments

Write a program using **`setup()`** and **`draw()`** functions. You **MAY NOT USE** GLOBAL VARIABLES. The program must do the following:

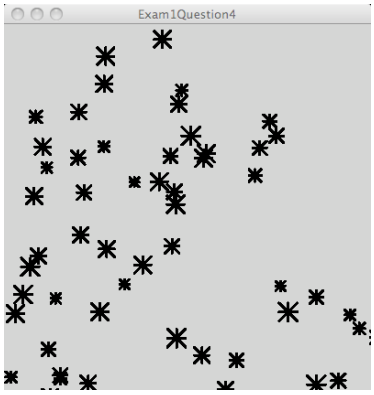
- define a function named **`figure()`** that draws the figure shown below




- The function must have three arguments:
  - two for the **`(x, y)`** coordinates
  - one for the size of the figure.
- The figure is always square.
- The anchor point location of the **`(x, y)`** coordinates are up to you. It can be a corner or the middle of the figure.
- The **`stroke`** and **`strokeWeight`** are the default values.
- One call of the figure( ) function results in ONLY ONE figure being drawn.
- The function **`draw()`** calls the **`figure()`** function using random numbers to position the figure and a reasonable random number for the size of the figure. Here is what your **`draw()`** function should look like:

```
void draw ( )
{
  figure(random(0, width), random(0, height), random(10, 20));
}
```

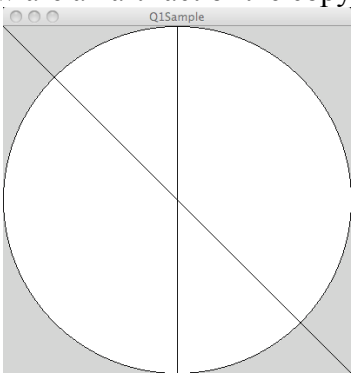
Here is a run of a sample solution:



# 257 Programming in the Arts with Processing
# Fall 2011 Exam 1 Sample for Practice

## Question 1 - 5 points

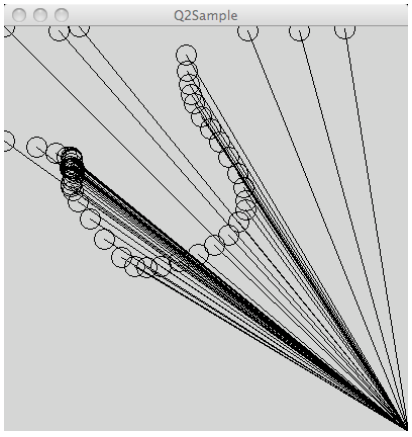
Write a program without setup( ) and draw( ) functions ( as you wrote the first few homeworks) and with no user input to draw the figure shown below. The code must use variables and expressions to locate the shapes. It must not use magic numbers. The figure must adjust to fit into different window sizes without any recoding other than changing the window size. The color of the background, the fill, and the stroke are the default colors – do not worry about them. The missing parts of the circle on the right and bottom sides of the figure below are an artifact of the copying and pasting within Word.




## Question 2 - 10 points

Write a program with a setup() and draw( ) function that does the following:

- draws a circle with no fill centered on the location of the mouse.
- has a diameter of the circle that is equal to 5% of the width of the screen.
- has a line must connecting the center of the circle with the lower right corner of the screen.
- has the background and stroke colors and stroke weight set to the default colors.

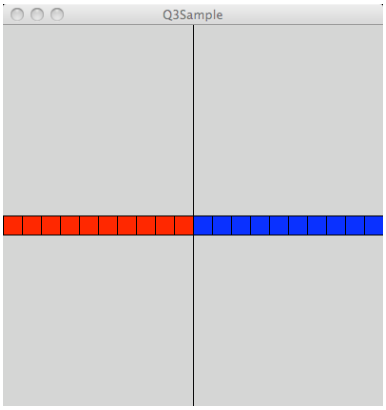


# Question 3 - 15 points

Write a program containing a setup() and a draw( ) function that does the following:

- draws a vertical line dividing the screen in half.
- fills all squares on the left side of the vertical dividing line with red ( 255, 0, 0 ).
- fills all squares on the right side o the vertical dividing line with blue (0, 0, 255);

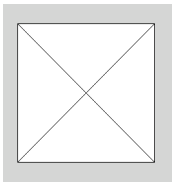
Here is one run of a sample solution:




# Question 4 - 20 points

Write a program containing a setup() and a draw( ) function that does the following:

- defines a function named figure( ) that has arguments to draw this figure at a location and size specified in the argument list of the call. The figure does not have to be square. The initial fill color of the figure is the default fill color:



- draws the figure at the mouse location when the user clicks the mouse.
- sets the size of the figure to a random value between 10 and 50 – it must not always be a square.
- erases the screen when the user presses the space bar.
- sets the fill color to a random color when the user presses any other key.

Here is one run of a sample solution: