IMAGE PROCESSING

Challenge: Instagram Filter Fill-in-the-Blank

```
//SET UP
PImage img; //Declare variable of type PImage

void setup() {
  size(200,200); //Change size of image.
  img = _____;
}

// CREATE IMAGE
void draw(){
_____;

// CREATE FILTER
_____;
 noStroke(); //no border
 rect(0,0,width,height); //Define shape of filter

}
```

IMAGE PROCESSING

Challenge: Instagram Filter Function Bank and Bonus Problem

image(img,0,0,width,height)	**Resize the image to the defined width and height (file name, x_origin, y_orgin, size x, size y)**
fill(255,255,0,60)	**Fill shape with semi-transparent filter over image (R value,G value,B value, alpha/ transparency)**
loadImage("IMG_3294.JPG")	**Download image from our library**

Bonus

Change RGB and alpha value of filter

Common Colors RGB values: http://www.workwithcolor.com/color-chart-full-01.htm

IMAGE PROCESSING

Challenge: Image Flip Fill-in-the-blank

```
//SET UP
PImage img, img_flip;
boolean ____________;

void setup() {
  size(750, 750);
  img =____________;
img_flip = createImage(750, 750, RGB);


_____________;
_____________;

//DEFINE FLIPPED IMAGE
 for (int i = 0; i < img.width; i++) { //i++ is iterating through the pixels horizontally
    for (int j = 0; j < img.height; j++) {
    img_flip.set(i, img_flip.height-1-j, _________);
    }
  }

  img_flip.updatePixels();

  __________ = false;
}
//DISPLAY IMAGE
void draw() {
  background(0);

  if (_______) {
    image(img_flip,0,0);
  }
  else {
    _____________;
  }
}

//CONDITION FOR MOUSE CLICK (USER INPUT)
void _____________{
  flip = !_____; //This symbol =! means not equal to
}
```

IMAGE PROCESSING

Challenge: Image Flip Function Bank

img.get(i, j)	**Reads the color of the specified pixel**
loadImage("spaceship.png")	**Download image from our library**
mouseClicked()	**Mouse clicked**
img_flip.loadPixels()	**Loads the pixel data for the image (img_flip) into its pixels array. This function must always be called be called before reading from or writing to pixelsbefore reading from or writing to pixels**
image(img,0,0);	**Display image (img) at origin (0,0)**
flip	**Whether image is flipped or not, value is True or False**
img.loadPixels()	**Loads the pixel data for the image (img) into its pixels array. This function must always be called be called before reading from or writing to pixelsbefore reading from or writing to pixels**

IMAGE PROCESSING

Challenge: Single Color Fill-in-the-blank

```
//SET UP
PImage img;
boolean _________;

void setup() {
_______________;
  img = loadImage("flowers.jpg");
_______________;
  _________ = false;
}
//DISPLAY IMAGE
void draw() {
  background(0);
  image(img, 0, 0);
}
//MOUSE CLICK, CHANGE COLOR
void mouseClicked() {
  single_color = !single_color;

  if (single_color) {
  float h = ___________________;
  _______________;
  //GET GREY SCALE EXCEPT FOR ONE COLOR
  for (int i = 0; i < img.width; i++) {
    for (int j = 0; j < _________; j++) {
      color c = __________;
      float ph = hue(c);
      if (abs(ph - h) > 10.) {
        img.set(i, j,____________);
      }
    }
  }
  img.updatePixels();
  }
  else {
    img = loadImage("flowers.jpg");
  }
}
```

IMAGE PROCESSING

Challenge: Single Color Function Bank

hue(get(mouseX, mouseY))	**Get the hue value of a pixel from a mouse click on the picture**
colorMode(HSB)	**Set the color mode to hue, saturation, and brightness**
img.loadPixels()	**Loads the pixel data for the image (img) into its pixels array. This function must always be called be called before reading from or writing to pixelsbefore reading from or writing to pixels**
img.height	**Number of pixels in the height of the image**
single_color	**Whether the picture is grey scale except for a single color. This value is True or False**
img.get(i,j)	**Reads the color of the specified pixel**
size(750, 500)	**Set size of the picture**
color(hue(c), 0, brightness(c)))	**Return the value of HSB: hue of "c" (the color of a pixel), 0 saturation (grey scale), and brightness of "c" (the brightness of a pixel).**

IMAGE PROCESSING

Challenge: Create a Vignette Fill-in-the-blank

```
//SET UP
PImage img, ________;
boolean vignette;

void setup() {
  size(460, 460);
  img = loadImage("inky.png");
  _________ = createImage(460, 460, RGB);
//CREATE A MASK
  msk.loadPixels();
  for (int i = 0; i < _________; i++) {
    for (int j = 0; j < msk.height; j++) {
        msk.set(i, j, color(255, 255, 255 - dist(i, j, width/2, height/2)));
    }
  }
  ____________;

  vignette = false;
}
//DISPLAY IMAGE
void draw() {
  background(0);
  image(img, 0, 0);
}
//CREATE VIGNETTE WHEN MOUSE IS CLICKED
void _____________ {
  vignette = !vignette;
  if (vignette) {
    ____________;
  }
  else {
    img = loadImage("inky.png");
  }
}
```

IMAGE PROCESSING

Challenge: Create a Vignette Function Bank

msk	**image mask file for the vignette**
msk.width	**Total number of pixels in the width direction of the msk image**
msk.updatePixels()	**Update pixels for the msk image**
img.mask(msk)	**Apply mask (predefined as msk) to image (img)**
mouseClicked()	**A function that is executed after the mouse button is pressed**