

Name: \_\_\_\_\_

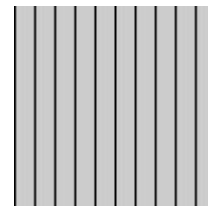
**CS 145 Images and Imagination**  
*Final Exam Solutions*

**Score:**

1. (max 8) \_\_\_\_\_
2. (max 6) \_\_\_\_\_
3. (max 8) \_\_\_\_\_
4. (max 25) \_\_\_\_\_
5. (max 10) \_\_\_\_\_
6. (max 8) \_\_\_\_\_
7. (max 8) \_\_\_\_\_
8. (max 12) \_\_\_\_\_
9. (max 6) \_\_\_\_\_

**Total: (max 91)** \_\_\_\_\_ **score scaled to 100:** \_\_\_\_\_

1. (8 pts) Write a for-loop to draw vertical lines spaced 10 pixels apart, as shown in the picture below.



```
for (int i = 0; i < width; i+=10) {  
    line(i,0,i,height);  
}
```

2. (2 pts each, 6 pts total) **Color:**

- What are the RGB values for bright blue? \_\_(0,0,255)\_\_
- What are the approximate RGB values for dark blue? \_\_(0,0, 100)\_\_
- What are the approximate RGB values for light, pale, blue? \_\_(100,100,255)\_\_

3. (4 pts each, 8 pts total) Suppose you want to draw random points in your window, where the color of a point is based on its location (i.e. x,y). To set the color, you use a conditional statement (e.g. if-else). What would this conditional statement have to be in order to set the color as follows.

- A point will have a blue (stroke) color if its (x,y) position falls in the *lower half* of the window, and red otherwise.

```
int x = random(width);
int y = random(height);
// add your code here:

if (y < height/2) {
    stroke(255,0,0);
}
else {
    stroke(0,0,255);
}
```

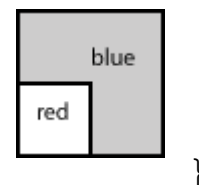


```
point(x,y);
```

- A point will have a red color if its position falls in the *lower left quadrant* of the window, and blue otherwise.

```
int x = random(width);
int y = random(height);
// add your code here:
```

```
if (y < height/2 || x > width/2) {
    stroke(255,0,0);
}
else {
    stroke(0,0,255);
}
```



```
point(x,y);
```

4. (5 pts each, 25 pts total) Complex numbers and imaginary numbers: Assume  $z_1 = 3 + 4i$  and  $z_2 = 1 - 2i$ .

Calculate the following. Place the result in standard form ( $a + bi$ )

a.  $z_1 + z_2 = \underline{4 + 2i}$

$$(3 + 4i) + (1 - 2i) = (3 + 1) + (4 - 2)i = 4 + 2i$$

b.  $z_1 * z_2 = \underline{\hspace{2cm}}$

$$(3 + 4i) * (1 - 2i) = \text{FOIL} = 3 - 6i + 4i - 8i^2 = 3 - 2i + 8 = 11 - 2i$$

c. Length of  $z_1 = |z_1| = \underline{\hspace{2cm}}$

$$|3 + 4i| = \sqrt{9 + 16} = \sqrt{25} = 5$$

d.  $i^4 = \underline{\hspace{2cm}}$

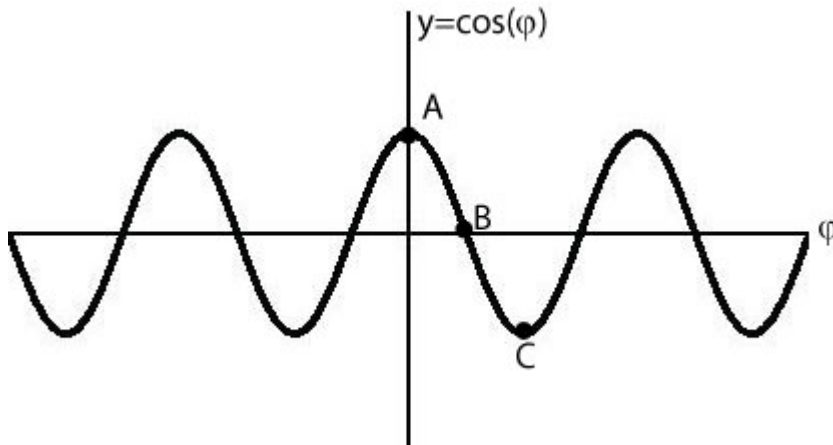
$$i^4 = i^2 i^2 = (-1)(-1) = 1$$

e.  $z_1 * i = \underline{\hspace{2cm}}$

$$(3 + 4i) i = -4 + 3i$$

5. (10 pts total) : Trig Values:

- a. (2 pts) The value of  $\sin(90^\circ)$  is   1
- b. (2 pts) The value of  $\cos(180^\circ)$  is   -1
- c. (6 pts) The graph of  $y = \cos(\theta)$  is given below (assume the angle  $\theta$  is in degrees):

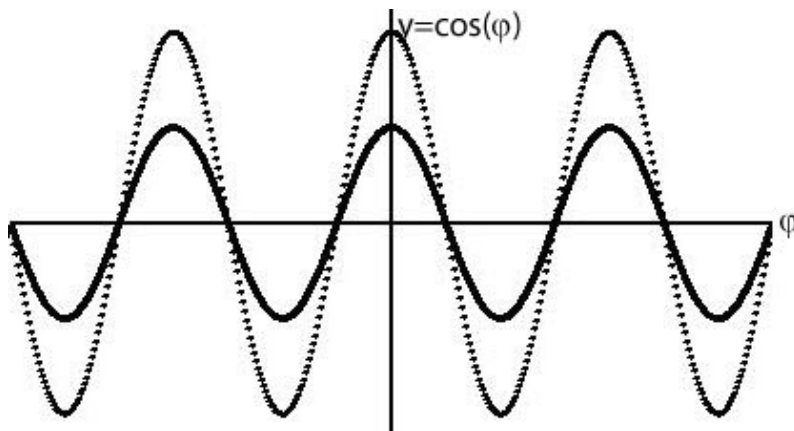


What are the coordinate values  $(\theta, y)$  at:

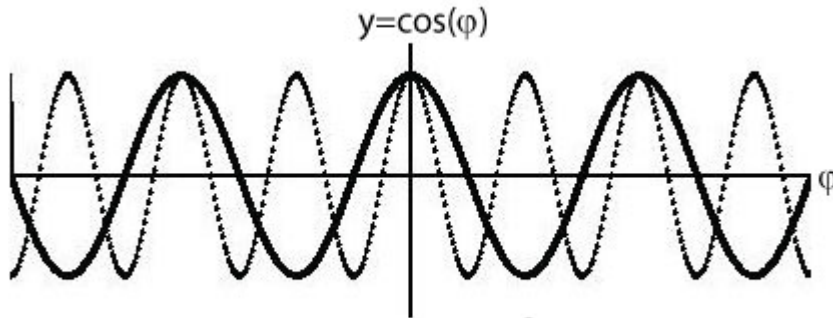
- i. Point A =  $(\theta, y) =$     $(0^\circ, 1)$
- ii. Point B =  $(\theta, y) =$     $(90^\circ, 0)$
- iii. Point C =  $(\theta, y) =$     $(180^\circ, -1)$

6. (8 pts total) **Trig Graphs:** The graph of  $y = \cos(\theta)$  is again given below.

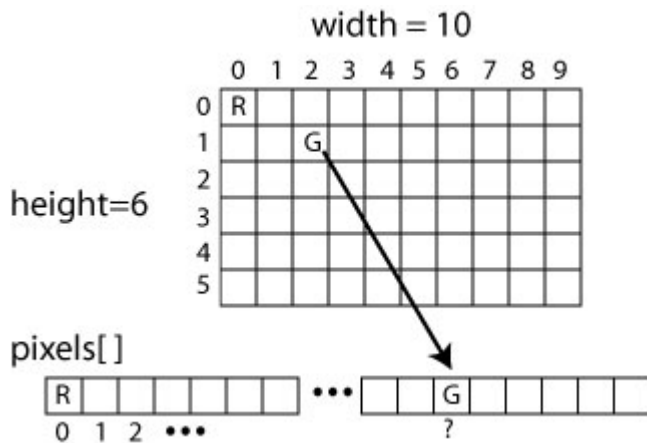
- a. Overlay on top of this graph, the graph for  $y = 2 * \cos(\theta)$ .



- b. The graph of  $y = \cos(\theta)$  is again given below.  
 Overlay on top of this graph, the graph for  $y = \cos(2*\theta)$



7. (8 pts total) **Image Filters:** Suppose we have an image that is 10 pixels wide and 6 pixels high as shown in the picture below. When one calls loadPixels() in Processing, the image's pixel colors will be loaded into the array called pixels[]. For example, the color (R) in the upper left corner of the image (i.e. 0<sup>th</sup> row and 0<sup>th</sup> column) will be placed in pixels[0] (i.e. the 0<sup>th</sup> index).



- a. Where will the color (G) in row 1 and column 2 be placed in the pixels array? That is, what will be its *index* in the pixels[] array?

index = 12

- b. In general, if a pixel is in row  $i$  and column  $j$ , what will be its index be in the pixels[] array?

$$\text{Index} = i * \text{width} + j$$

8. (4 pts each, 12 pts total) Given the Processing program:

```
void setup() {
  size(100,100);


  // transformations go here
  house();
}
```




The above code, without transformations, will generate the above picture. For each of the following set of transformations, determine which picture will be drawn if the transformations are placed in the location indicated in the above program.

a. `translate(width/2, height/2);`  
`rotate(radians(45));`

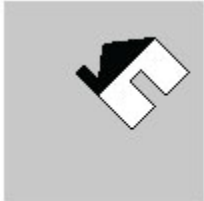
*Circle the correct answer*



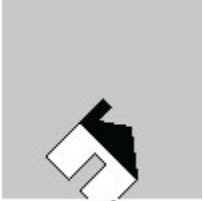
a



b



c




d


↑ ans

b. `scale(2);`


*Circle the correct answer*




a



b



c




d


↑ ans

c. `translate(width/2, height/2);`


*Circle the correct answer*




a



b



c

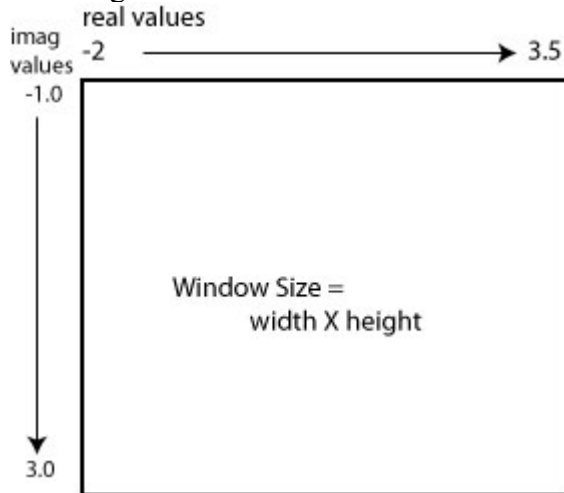


d

↑ ans

9. (6 pts) (pts) **The Map Function:** Recall that in the Mandelbrot lab, we had to take a pixel location (i,j) and determine it's corresponding value in the complex plane.

Given a region of the complex plane where the real and imaginary components range as shown in the figure below:



How does one use the map function to determine the pixel location of the complex number  $z = -0.4 + 1.5i$ .

```
int pixeli = map(-0.4, -2, 3.5, 0, width);
```

```
int pixelj = map( 1.5, -1, 3.0, 0, height);
```