

Name: _____

CS 145 Images and Imagination*Exam 2***Score:**

1. (max 28) _____

2. (max 24) _____

3. (max 24) _____

4. (max 24) _____

Total: (max 100) _____1. (28 pts total) **Arrays:**

- a. (5 pts) Write code below to declare and create an array (e.g. called nums) of 50 integers.

```
int nums[] = new int[50];
```

- b. (6 pts) Below, write a loop (e.g. which would be placed in setup()) which initializes the value of each element in the array to be equal to 10 times it's index. For example, the first element has index 0, so its value would be 0. The next element has index 1, so its value would be 10, the next element has index 2 so its value would be 20, etc.

```
for (int i=0; i < nums.length; i++) {
    nums[i] = 10*i;
}
```

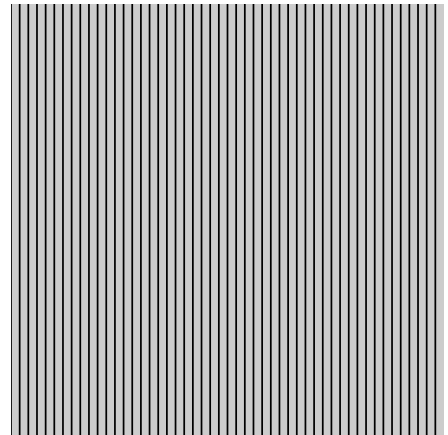
- c. (5 pts) String concatenation: What code could be added to the above loop to print out the value of each element in the array? The resulting output should look like:

```
i = 0  nums = 0
i = 1  nums = 10
i = 2  nums = 20
etc, on down to
i = 49  nums = 490
```

```
println("i = " + i + " nums = " + nums[i]);
```

- d. (6 pts) Write a second loop that draws 50 vertical lines (as shown here) whose x positions are given by the values in the nums array.

```
for (int i=0; i < nums.length;i++) {
    line(nums[i],0,nums[i],height);
}
```



- e. (6 pts) Write a third loop that adds up the value of all of the elements in the array and prints the result after the loop is done.

```
int sum = 0;
for (int i=0; i < nums.length;i++) {
    sum = sum + nums[i];
}
```

2. (6 pts each, 24 pts total) Given the program below, what does the matrix stack contain at the indicated lines?

```

void setup() {
  size(200,200);
  translate(width/2,height/2);
Line 1  rect(0,0,10,20);
        pushMatrix();
        scale(1,-1);
Line 2  rect(0,0,10,20);
        popMatrix();
        translate(20,10);
Line 3  rect(0,0,10,20);
        for (int i = 0; i < 3; i++) {
          rotate(radians(10*i));
          rect(0,0,10,20);
        }
Line 4  rect(0,0,10,20);
        }

```

- a. Line 1:

```
translate(width/2,height/2)
```

- b. Line 2:

```
scale(1,-1)
translate(width/2,height/2)
```

- c. Line 3:

```
translate(20,10)
translate(width/2,height/2)
```

- d. Line 4:

```
rotate(radians(20))
rotate(radians(10))
rotate(radians(0))
translate(20,10)
translate(width/2,height/2)
```

3. (3 pts each, 24 pts total) Given the Processing program:

```

void setup() {
  size(100,100);
  rectMode(CENTER);

  // transformations go here
  rect(0,0,10,40);
}

```



The above code, without transformations, will generate the above picture. Note, only a quarter of the rectangle can be seen. 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. Each set of transformations has a match to an image, however, not all images have matches to transformations.

a. `rotate(radians(90));` | **ans: 3**

b. `translate(width/2, height/2);`
ans: 9

c. `translate(width/2, height/2);`
`rotate(radians(90));` | **ans: 4**

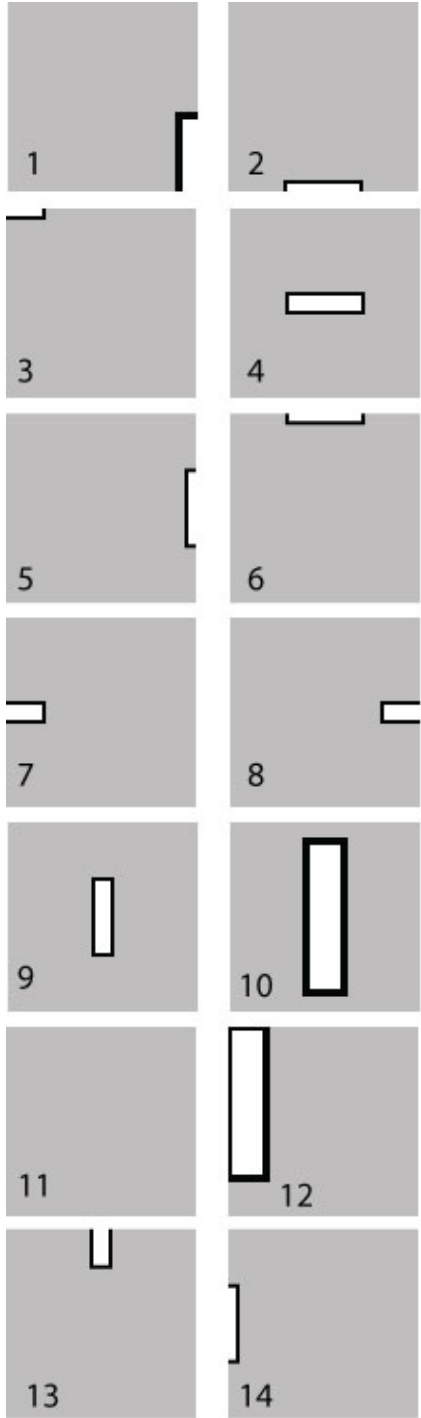
d. `rotate(radians(90));`
`translate(width/2, 0);`
ans: 7

e. `rotate(radians(-90));`
`translate(0, height/2);`
ans: 6

f. `scale(2);`
`translate(width/2, height/2);`
ans: 1

g. `translate(width/2, height);`
`rotate(radians(-90));`
`translate(width/2, height/2);`
ans: 8

h. `scale(2);`
`translate(5, 20);`
`rect(0,0,10,40);`
ans: 12



4. (8 pts each, 24 pts total) The code below draws a foot image in the upper left corner of the window as shown.

```
PImage foot;
void setup() {
  size(100,100);
  foot = loadImage("foot.gif");
  image(foot,0,0);
}
```



- a. (8 pts) Write a *function* called `reverseFoot` which draws the foot image in the upper left corner but reflected as shown.

```
void reverseFoot() {
  pushMatrix();
  translate(foot.width,0);
  scale(-1,1);
  image(foot,0,0);
  popMatrix();
}
```



Reversed Foot

- b. (8 pts) Write a *second function* called `basePattern` that makes use of the `reverseFoot` function in order to generate the image shown here on the right.

```
void basePattern() {
  pushMatrix();
  image(foot,0,0);
  translate(foot.width,0);
  reverseFoot();
  popMatrix();
}
```



- c. (8 pts) In the `setup()` function given below, write a loop which generates the frieze pattern shown below (assume you have your `basePattern` and `reverseFoot` functions available).



```
void setup() {
    foot = loadImage("foot.gif");
    size(8*foot.width,foot.height);

    // add code below

    for (int i = 0; i < 4;i++) {
        basePattern();
        translate(2*foot.width,0);
    }

}
```

***** OR *****

```
for (int i = 0; i < 4;i++) {
    pushMatrix();
    translate(2*foot.width*i,0);
    basePattern();
    popMatrix();
}
```