CS101
Spring 2018

Instructor: Ashley Taylor
June 8, 2018

Final Exam

NAME (LAST, FIRST): ______________________________

SUNET: ______________________________@stanford.edu

Problem	1	2	3	4	5	6	TOTAL
Score							
	21	9	12	30	20	25	117

Instructions:

- The time for this exam is **3 hours**, 180 minutes total. There are 117 points total, so you should spend a little more than 1.5 minutes per point, though your pacing may be different on different problems.
- You are only allowed a pencil, pen, and eraser. Any other materials are strictly prohibited. You may not use any digital devices other than a clock for timing. You may ask the proctors for blank scratch paper.
- A **reference sheet** is included as the last page. You may detach it for your reference during the exam.
- For coding problems, we will forgive minor syntax errors (such as missing semicolons, etc.).
- For any problems involving calculations, we will accept an expression (that could be plugged into a calculator) in lieu of the numerical answer.
- Problem 4b is overweighted by points compared to the time it will take. That might be a good place to start.

Please sign *before* beginning the exam:

I agree to abide by the spirit and letter of the Honor Code, and to follow the instructions above.

______________________________ ______________

(Signature) (Date)

1. **Course Themes**

   a) (6 points) Name two different examples of shared resources in computer science. For each example, describe how that resource is managed.

   A non-CS example of a shared resource would be the washing machines in your dorm, which are managed by RAs (who have authority to instruct the users on how to use them) and the mailing list (which uses public shame to encourage the desired behavior).

   b) (4 points) Describe one area in computer science where standardization is effectively implemented, and describe why standardization was necessary. Describe another area where standardization was not completely implemented; what is the impact of the lack of standardization in that case?

c) (8 points) In class, we discussed layers of abstraction in regard to two different systems. What are these two systems? For each system, which layer is the most abstract, which layer is the least abstract, and describe the purpose of one of the layers.

d) (3 points) What is a graph? What are two different graphs we described in class?

2. **Web and the Internet**
    a) (2 points) Why do changes to CSS through Google Chrome inspector, as we did in class, only affect how the website looks on your computer, but not on everyone else's computer?

    b) (4 points) Describe the three frontend components. How do they work together to display webpages?

    c) (3 points) Name 3 factors that influence the ordering of search results in Google?

3. **Security**
   a) (4 points) What is public-key encryption? Describe two uses for public-key encryption.

   b) (2 points) What is the difference between security and privacy?

c) (2 points) What is a password manager? How does it make your passwords stronger?

d) (4 points) Describe two different kinds of attacks (or “hacks”). How can either computer scientists or users protect against each type?

4. **Potpourri**
   a) (3 points) Describe a dark pattern. How could you as a user subvert this pattern?

   b) (10 points) What is a checksum? In what context is it used, and why is it necessary? Note: this question is overweighted by points compared to how long it will take you to complete. 3-4 sentences should suffice.

c) (5 points) Describe how the code you write is understood by a computer. What steps are involved?

d) (2 points) What is the analogous element in JavaScript to a single cell in a spreadsheet? Name one other similarity between spreadsheets and JavaScript.

e) (4 points) Describe the P vs. NP question. Name two implications if P equals NP?

f) (4 points) Describe two types of data that are stored in computers. How is each type stored?

g) (2 points) What is a design change you would make to the CS101 website?

5. **Code Reading/Comprehension**

   a) (8 points) Below is the code from edge detection as we talked about in class. Edit it so that edges are colored blue and non-edge pixels are colored yellow. Additionally, we will add the constraint that an edge cannot be on the left border of the image, so all left border pixels should be considered non-edges. You can cross out or add code as necessary, but please use arrows to clearly indicate your edits. Hint: you don't need to edit any of the first four lines of code.

```
image = new SimpleImage("striped-fish-green.jpg");
back = new SimpleImage("yosemite.jpg");
back.setSameSize(image);
back.setZoom(0.5);

for (pixel : back) {

  pixel.setRGB(255, 255, 255);

}

for (pixel : image) {

  for (neighbor : image.getNeighbors(pixel)) {

    red = neighbor.getRed();
    green = neighbor.getGreen();
    blue = neighbor.getBlue();

    if (!pixel.isSimilarTo(red, green, blue, 20)) {

      backPixel = back.getPixel(pixel.getX(), pixel.getY());

      backPixel.setRGB(255, 255, 255);

    }

  }

}
print(back);
```

b) (6 points) Below is a snippet of code with several bugs. It attempts to increase contrast in red by rescaling the red values (for example, if an image only had red values between 50 and 100, we could increase the separation by scaling the pixels with a red value of 50 to 0, those with a red value of 100 to 255, and the rest would be in between). Identify and correct one syntax error and two logic errors, clearly identifying which is which:

```
image = new SimpleImage("5b.jpg");
maxRed = 0;
minRed = 0;
for (pixel : image) {
    if (pixel.getRed() < maxRed) {
        maxRed = pixel.getRed();
    } else if (pixel.getRed() < minRed) {
        minRed = pixel.getRed();
    }

for (pixel : image) {
    pixel.setRed(getRed() - minRed * 255 / (maxRed - minRed));
}
print(image);
```

c) (6 points) Below are the first two rows of a sample spreadsheet. Describe how you would determine which team has the highest average number of players per year (note: you do not need to use any commands, though you may do so if those would help you explain). Your description should be detailed enough that a novice Excel user could determine which functions in which order they should use.

Year	Team	Number of Players	Number of Points
2001	Giants	31	139
...	...	...	...

6. **Code Writing**

   a) (6 points) Color the 3x3 square at the bottom-left corner of an image yellow. Your solution must use a for loop.

```
img = new SimpleImage("6a.png");
```

```
print(img);
```

b) (7 points) We learned about steganography, a way of hiding messages in plain sight. You will write code to unveil a hidden image inside another image.

One way of hiding an image is to slightly modify the green value of a pixel - an odd green pixel in the original image corresponds to a black pixel in the hidden image, and an even green value in the original image corresponds to a white pixel in the hidden image. You should write code that converts the original image to the hidden image.

You can use the code building block `isOdd(value)`, which is `true` if value is odd.

```
img = new SimpleImage("6b.png");
```

```
print(img);
```

c) (7 points) Make the following image entirely grayscale, except for a purple diagonal offset by a height of 2 (from the bottom left corner to the top right). For example, the pixels marked by an X below should be purple:

		x
	x	
x		

```
img = new SimpleImage("6c.png");
```

```
print(img);
```

d) (5 points) Write code to flip an image horizontally, as shown in the example below. Note: this question is underweighted by points compared to how long it will take you to complete. You should work on it last.

`example.png`

red	green	blue
blue	white	red
green	blue	black

`expected output`

blue	green	red
red	white	blue
black	blue	green

```
img = new SimpleImage("6d.png");
```

```
print(img);
```

This page is intentionally left blank.

```
pixel.getX()
pixel.getY()
pixel.getRed()
pixel.getGreen()
pixel.getBlue()
pixel.setRGB(red, green, blue)
pixel.isSimilarTo(red, green, blue, threshold)

image = new SimpleImage("image.png")
image.countNeighbors(pixel)
image.getHeight()
image.getWidth()
image.getPixel(row, column)

for (pixel : image) {
   // your code here
}

for (neighbor : image.getNeighbors(pixel)) {
  // your code here
}

if (condition) {
  // your code here
} else {
  // your code here
}

&& => and
|| => or
! => not
!= => is not equal
== => is equal

1TB = 1000GB
1GB = 1000MB
1MB = 1000KB
1KB = 1000 bytes

Bit: 0 is "off", 1 is "on"
```