CS101 Instructor: Ashley Taylor
Spring 2018 May 10, 2018

Midterm Exam

NAME (LAST, FIRST): _____

SUNET: _____@stanford.edu

Problem	1	2	3	4	5	TOTAL
Score						
	9	9	10	21	16	65

Instructions:

- The time for this exam is **1 hour and 20 minutes**, 80 minutes total. There are 65 points total, so you should spend a little more than 1 minute 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.

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. **Data Storage**

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

Many possible answers. An example answer: Integers and characters are two types of data stored in computers. Integers are stored as a binary representation of the number. Characters are typically stored in ASCII representation.

b) (3 points) Each text message contains a date (which is 8 bytes), a time sent (which is 4 bytes), up to 160 characters of text, and the phone number of the sender (which is an 8-byte number). How many text messages can you store on an 128GB hard drive?

Each text message will need a date (8 bytes), a time sent (4 bytes), text (up to 160 characters), and a phone number (8 bytes). If we assume each character is stored in ASCII representation, then each character will take 1 byte. So each text will need 8 + 4 + 160 + 8 = 180 bytes. 128GB = 128,000MB = 128,000,000KB = 128,000,000,000Bytes. So we could fit 128,000,000,000 Bytes / 180 Bytes per text. That gives 711,111,111 texts.

c) (2 points) What would happen if you ran out of RAM? What would happen if you ran out of permanent storage?

RAM is the computer's temporary workspace. If you ran out of RAM, your programs wouldn't be able to run. Most likely your computer would slow down tremendously if you ran out of RAM.
If you ran out of permanent storage, your computer would no longer be able to permanently store any of its files. You would no longer be able to save any work.

---

2. **Hardware**

a) (2 points) Rank the following from most to least abstract, where 1 is the most abstract:

2	Programming Language
6	Transistors
3	Operating System
4	Architecture
1	Applications
5	Computer Components (CPU, Permanent Storage, and RAM)

b) (3 points) Choose three layers of abstraction (from the list in part a). Describe the purpose of each layer.

Programming Language - Allows programmers to code significant projects without worrying about how a computer actually works, they can just use a high-level language that resembles english.
Transistors - These are the actual bits in the computer, the 1s and 0s.
Operating System - Manages your computer, programs, RAM, and persistent storage. Starts, stops, and runs programs and shares memory between them.
Architecture - This is the link between hardware and software. The machine code is instructions in 1s and 0s that the computer will understand.
Applications - The programs that actually run on your computer. This is why computers are pervasive. It abstracts all the details away and allows a person who has no idea how a computer works to browse the internet or play music.
Computer Components - Groups the hardware into separate components with separate functions.

c) (4 points) Below is information about two different computers:

Computer A	Computer B
128GB Flash Drive	512GB Hard Drive
16GB RAM	8GB RAM
150DPI Screen	225DPI Screen
15 inch screen	13 inch screen

Assume that the costs of the two computers are the same. List two strengths of Computer A and two strengths of Computer B, describing the impact of each strength (as in, the sorts of tasks that it might be better at as a result). An example has been done for you:

Computer A has a 15 inch screen, whereas Computer B has a 13 inch screen. As a result, more information can be displayed on Computer A's screen at a time, leading to better multi-tasking.

Computer A strengths:

1. Computer A has a flash drive instead of a hard drive. Flash is faster, has no moving parts, and requires less power.
2. Computer A has 16GB of RAM. This will allow you to do more with Computer A since you have a larger temporary workspace.

Computer B strengths:

1. 512GB of persistent storage. You will be able to store a lot more files on this computer.
2. 225DPI Screen has a higher resolution, so watching movies, looking at pictures, or other visual media will look better on computer B.

---

3. Code Reading Questions

a) (3 points) What is the value of x at the end of this code? Assume that 3pixels.jpg has 3 pixels. Show your work for partial credit.

```
x = 3;
y = 7;
img = new SimpleImage("3pixels.jpg");
for (pixel : img) {
    x = x + y;
    x = x + 1;
    y = 1;
}
```

The for loop runs 3 times. After the first run through the for loop, x=11, y=1. After the second run through the for loop, x=13, y=1. After the final run through the for loop, x=15, y=1. So x=15 at the end of this code.

b) (3 points) The following code is intended to change the top and bottom rows of x.png to orange. It has two logic bugs and three syntax bugs. Identify and correct **one** logic error and **two** syntax errors, clearly stating which is which.

```
img = new SimpleImage("x.png");
for (pixel : img) {
    if (pixel.getY() == 0 && pixel.getY == img.getHeight() - 1) {
        setRGB(0, 255, 125);
}
print(img);
```

Syntax Bugs:

1. The second call to pixel.getY() is missing parentheses
2. The call to setRGB should be: pixel.setRGB(0, 255, 125);
3. The if statement is missing the ending curly brace '}'. That should be inserted after the call to setRGB.

Logic Bugs:

1. The call to setRGB(0, 255, 125) has mixed up the arguments. It should be: setRGB(255, 125, 0) for orange.
2. The if statement should use OR, not AND. Change the '&&' to '||'.

c) (4 points) Draw what the output of the following code is, clearly labeling each pixel with its color (each box is a pixel); use English colors (e.g. black) instead of RGB values. Assume that isOdd(number) is true if number is an odd number.

```
img = new SimpleImage("output.png");
for (pixel : img) {
    if (isOdd(pixel.getX() + pixel.getY())) {
        pixel.setRGB(255, 0, 255);
    } else {
        pixel.setRGB(255, 0, 0);
    }
}
print(img);
```

The boxes below are labeled: “x+y -> Color”. Note colors resembling magenta, violet, or other similar colors were accepted as answers.

0+0 -> Red	1+0 -> Purple	2+0 -> Red
0+1 -> Purple	1+1 -> Red	2+1 -> Purple
0+2 -> Red	1+2 -> Purple	2+2 -> Red

---

4. Code and Spreadsheets

a) (3 points) How would you store a JavaScript pixel in a spreadsheet? Clearly indicate what columns you would have, and what each row would represent. For reference, pixels have the following JavaScript methods to get information about the pixel - the same information should be accessible for a pixel in your spreadsheet.
Note: **you should not write code for this part**, but drawings of your spreadsheet are encouraged.

```
pixel.getX()
pixel.getY()
pixel.getRed()
pixel.getGreen()
pixel.getBlue()
```

Every row would represent a different pixel. Each pixel is made up of 5 pieces of information (X, Y, red, green, blue) so these would be our columns. That would look something like this:

X position	Y position	red	green	blue
0	0	255	0	0
1	0	120	16	201

Where the numbers in the cells are chosen arbitrarily for this example.

b) (10 points) Now, we're going to write code to put a JavaScript image into a spreadsheet. The result is that we should have every pixel in the original JavaScript image stored correctly in the spreadsheet. We've provided some code below that will open the image and spreadsheet.

We've also provided the following new code building blocks:
`spreadsheet.setColumns("columnName1", "columnName2");`
include however many column names you have, similar to print

`nextRow = spreadsheet.getNextRow();`
nextRow is a number representing the next empty row number in the spreadsheet

`spreadsheet.setCell(row, "columnName", value);`
row should be a number (see above), columnName should be the name of your column, and value is any value that you want to store in the spreadsheet. As an example, `spreadsheet.setCell(nextRow, "age", 5);` would store 5 in the age column of the next row.

Hint: the next line of code should be `spreadsheet.setColumns(...);`

Write your answer on the next page.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Your code for part b below:

```
img = new SimpleImage("myimage.jpg");
spreadsheet = new Spreadsheet("partBResult.csv");
spreadsheet.setColumns("X", "Y", "Red", "Green", "Blue");
for (pixel : img) {
      rowNumber = spreadsheet.getNextRow();
      spreadsheet.setCell(rowNumber, "X", pixel.getX());
      spreadsheet.setCell(rowNumber, "Y", pixel.getY());
      spreadsheet.setCell(rowNumber, "Red", pixel.getRed());
      spreadsheet.setCell(rowNumber, "Green", pixel.getGreen());
      spreadsheet.setCell(rowNumber, "Blue", pixel.getBlue());
}
```

c) (8 points) For this part, we're going to make an image grayscale **in the spreadsheet**. You should set the cell values directly, rather than trying to convert from the spreadsheet to an image. You can assume that the spreadsheet is the same as from part b (so it has the same column names). We've provided the following code building blocks:

`spreadsheet.getCell(row, "columnName");`
Gives back the value in a cell at the given row (which is a number) and columnName (opposite of setCell from part b)

`for (row : spreadsheet) { ...`
Goes over all the rows in the spreadsheet. Row takes on the values of the used row in the sheet (i.e. if there were 100 rows in the sheet, row would first be 1, then 2, and so on up to 100). `row` can be used as the row argument for `getCell` and `setCell`

```
spreadsheet = new Spreadsheet("partBResult.csv");
for (row : spreadsheet) {
      red = spreadsheet.getCell(row, "Red");
      green = spreadsheet.getCell(row, "Green");
      blue = spreadsheet.getCell(row, "Blue");
      brightness = (red + green + blue) / 3;
      spreadsheet.setCell(row, "Red", brightness);
      spreadsheet.setCell(row, "Green", brightness);
      spreadsheet.setCell(row, "Blue", brightness);
}
```

---

5. Potpourri

a) (2 points) Facebook has had problems with explicit content in Live Videos uploaded to the site. How could they use Artificial Intelligence to confront this problem? Which kind(s) of AI that we talked about in class (natural language processing, artificial vision, and robotics) could be used?

Artificial vision would be useful here. Using artificial vision, facebook would be able to tag live videos if artificial vision determined that there was some sort of explicit content in the videos.

We would also accept natural language processing (processing the audio of the videos searching for explicit content).

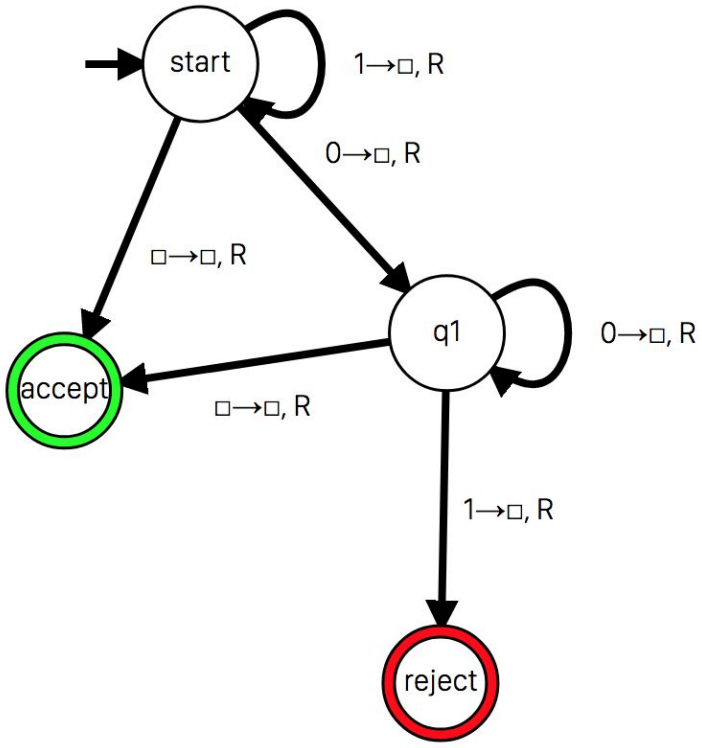
b) (2 points) Describe open-source software. How is it different from traditional software, and why would users or software developers prefer open-source?

Open-source software is software made free to use, and free to edit to the general public. Traditional proprietary software cannot be viewed or edited by anyone outside of the company that developed it. Users and software developers prefer open-source because it can lead to a higher quality product if it has many different contributors. Many also prefer the ideology of open-source software, believing that software should be open to all.

c) (4 points) Describe two ways a program might end. What abstraction layers of a computer are involved in each?

A program could end normally if the user quits that program. In this case, the operating system would stop the program and reclaim the program's area of RAM. If the program stops responding, then the operating system can forcefully stop that program and reclaim the program's area of RAM. So the operating system as well as computer components (RAM) are the abstraction layers involved.

d) (4 points) Does the Turing Machine below accept or reject the input 1001? What is one input it accepts and one it rejects (besides 1001)? Each input should at least start with a number and be at least two non-blank characters.



This Turing machine rejects the input 1001.
Inputs that it accepts:
111111
111
10000000
10
11110000

Inputs that it rejects:
101
100001
01
000001

e) (2 points) How does a digital camera work?

The lens focuses light. The CCD (charged coupled device) records light in a grid of cells. Each of these cells is a pixel. Each pixel has some amount of red, green, and blue saved for that pixel.

f) (2 points) What is a tradeoff and a benefit of data compression?

Data compression allows you to store data using less storage (i.e. you can fit more data onto your computer). However compression can lower quality, and also can take more time to 'translate' the data back to its original state if you are using an algorithm like Huffman Encoding.

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"
```