Midterm Exam

NAME (LAST, FIRST): ____________________________________________________________

SUNET: ________________________________________________________@stanford.edu

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>21</td>
<td>16</td>
<td>65</td>
</tr>
</tbody>
</table>

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?

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?

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

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

   1. Programming Language
   2. Transistors
   3. Operating System
   4. Architecture
   5. Applications
   6. 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.
c) (4 points) Below is information about two different computers:

<table>
<thead>
<tr>
<th>Computer A</th>
<th>Computer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>128GB Flash Drive</td>
<td>512GB Hard Drive</td>
</tr>
<tr>
<td>16GB RAM</td>
<td>8GB RAM</td>
</tr>
<tr>
<td>150DPI Screen</td>
<td>225DPI Screen</td>
</tr>
<tr>
<td>15 inch screen</td>
<td>13 inch screen</td>
</tr>
</tbody>
</table>

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.
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.

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

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.

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

```java
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);
```

![Output Image](attachment:output.png)
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()
```
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();
extRow 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.
Your code for part b below:

```java
img = new SimpleImage("myimage.jpg");
spreadsheet = new Spreadsheet("partBResult.csv");
```
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:

```java
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)

```java
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

```java
spreadsheet = new Spreadsheet("partBResult.csv");
```
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?

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

c) (4 points) Describe two ways a program might end. What abstraction layers of a computer are involved in each?
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.
e) (2 points) How does a digital camera work?

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