Math 230 — Final Exam

May 12, 2015

General Directions. This is an open-book, open-notes, open-computer test. However, you may not communicate with any person, except me, during the test. You have the full exam period (3 hours) in which to do the test. Put your answer to each question in the space provided (use the backs of pages if you need more space). Be sure to **show your work**! I give partial credit for incorrect answers if you show correct steps leading up to them; conversely, I do not give full credit even for correct answers if it is not clear that you understand where those answers come from. Good luck.

This test contains 7 questions (2 with two parts) on 8 pages.

Question 1 (10 Points). Somewhat surprisingly, the sum

$$\sum_{k=1}^{n} \frac{1}{k}$$

grows without bound as *n* increases. It does, however, grow very slowly. Here is the header for a Matlab function that takes a bound, *x*, as its argument, and that is supposed to return the smallest *n* such that the above sum evaluated from *k* = 1 to *k* = *n* is greater than *x*. Fill in the function's body.

```
function [ n ] = findN( x )
```

Question 2 (20 Points). Imagine that you want to solve inequalities of the form

$$xe^{x} \geq c$$

where c is a constant that varies from instance to instance of the inequality. Explain how you could use bisection to do this. In other words, write a combination of prose and pseudocode to show how you could use the idea of bisecting an interval according to how some function's value compares to a target value to find the smallest *x* that solves the above equation, given the value of *c*.

I can imagine several ways of answering this question, some of which mainly involve writing a modified form of the bisection algorithm we discussed in class, and others of which involve using the unmodified algorithm, but in a way that tells you what the solution to the inequality is. You may use either approach, or others that you find; if you use an unmodified version of bisection as presented in class you do not need to write all of its code down, just make it clear in your discussion that you are using the class version of the algorithm.

Question 3 (10 Points). Imagine that file "message.mat" contains a single Matlab variable named `msg`. This variable holds a string. Write a Matlab script that does the following:

1. Load variable `msg` from "message.mat"
2. Convert the string contained in `msg` to uppercase
3. Save the converted string, and no other variables or values, into a new file named "upper.mat"

Question 4. An illustrated book can be thought of as a sequence of text blocks and images, which alternate (i.e., first there is a text block, then an image, then another text block, then an image, and so forth). Assume that the sequence begins and ends with text blocks.

Here is one rather awkward way to store an illustrated book in a one-dimensional cell array: each cell contains either (1) a single character of text, as a string, (2) the multi-character string "IMAGE COMING UP," or (3) an image, as an *n*-by-*m*-by-3 array of numbers, just like in our chroma key lab. The individual characters spell out the text parts of the book, "IMAGE COMING UP" signals that the next cell in the cell array holds an image, and the images are hopefully self-explanatory.

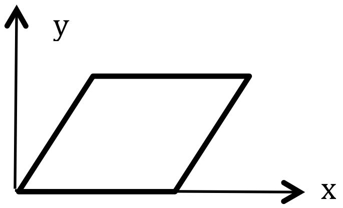
Part A (10 Points). Describe a simpler way to represent an illustrated book as a cell array. Your cell array can have as many dimensions as you want.

Part B (15 Points). Write a Matlab function that takes an illustrated book as represented in Part A (or using my representation if you prefer), and displays it. In other words, your function should print the text parts of the book, and display the images in one or more figure windows. Your function should have no result variable.

Question 5 (10 Points). Define a hyphenated phrase to be a string containing words separated by hyphens. For example “final-exam-season” and “hyphen-misuse” are both hyphenated phrases.

Write a Matlab function that takes a string containing a hyphenated phrase as its argument, and that returns the number of words in that phrase. Assume that there is exactly one hyphen between every pair of words in the phrase (e.g., your function will never receive strings such as “hyphenated--phrase”).

Question 6. Polygons can be represented by two vectors, one containing *x* coordinates of the polygon's vertices and the other containing *y* coordinates. For example, suppose this polygon



has vertices (0, 0), (0.5, 1), (1.5, 1), and (1, 0). It could be described by the vectors (0, 0.5, 1.5, 1) containing the *x* coordinates and (0, 1, 1, 0) containing the *y*. In this representation of polygons, consecutive coordinates in the vectors describe adjacent vertices of the polygon (e.g., in the above example, vertices (0, 0) and (0.5, 1) are adjacent—a side of the polygon connects them—and are consecutive in the vectors—namely, the first elements define (0, 0) and the second define (0.5, 1)).

Part A (15 Points). Here is the header for a Matlab function that takes vectors of *x* and *y* coordinates for a polygon as described above, and calculates the length of the perimeter of the polygon. Fill in the body of the function using only loops and no vectorized operations. (Warning: don't forget the side from the last vertex back to the first.)

```
function [ p ] = perimeterLoop( x, y )
```

(Part B on next page)

Question 6, Part B (20 Points). Here is the header for a Matlab function that takes vectors of *x* and *y* coordinates for a polygon as described above, and calculates the length of the perimeter of the polygon. Fill in the body of the function using only vectorized operations and no loops. (Warning: don't forget the side from the last vertex back to the first.)

```
    function [ p ] = perimeterVectorized( x, y )
```

Question 7 (10 Points). Consider the mathematical function *f* defined as follows:

$$f(x) = \begin{array}{c} (x+1)^2 \text{ if } -1 \le x \le 0 \\ 1 - x \text{ if } 0 < x \le 1 \\ 0 \text{ otherwise} \end{array}$$

Here is the header of a Matlab function that takes *x* as its parameter and computes *f*(*x*). Fill in the body of this function.

```
function [ y ] = f( x )
```

Have a Good Summer!