Problem Set 7

Due Date: Sat Nov 14, 2015
by 9 PM

Name:________________________________

Lab Section & TF:______________________

Collaborators:_________________________

For Grading Purposes Only:

Q1: _____ / 20
Q2: _____ / 25
Q3: _____ / 10
Q4: _____ / 10
Q6: _____ / 16
Q7: _____ / 26
BONUS: _____ / 12

Total:_____ / 107 + 12

Problem 1: Practical Combinational Logic (20 points)

A comparator is a circuit that compares two numbers, A and B, and generates two output signals, E and G, according to the following rules:

- G = 1 if A>B
- G = 0 if A<B
- G = X if A=B (Note: X = don't care; you can choose it to be whichever value, 1 or 0, you want)
- E = 1 if A=B
- E = 0 otherwise

a) (5 points) Write out the Boolean expression for a 1-bit comparator, which compares two 1-bit numbers A and B, and generates two 1-bit outputs, E and G, corresponding to the rules above.

b) (5 points) Only using 2-input inverting logic gates (e.g., NAND, NOR, XNOR) and inverters, draw the circuit that implements the 1-bit comparator. You can assume both true and compliments of A and B are available, i.e., A, A', B, and B' available.

c) (5 points) Now, write out the Boolean expression for a 2-bit comparator, which compares two 2-bit numbers $A=A_1A_0$ and $B=B_1B_0$, and generates two 1-bit outputs, E and G.

d) (5 points) Draw the circuit for the 2-bit comparator only using 2-input inverting logic gates and inverters.

Problem 2: Ripple-Carry Adders (25 points)

In Lecture 11, we saw how to create circuits that can add two numbers together by cascading multiple instances of 1-bit full adders (FA). This problem delves a bit deeper into understanding multi-bit adders.

a) (5 points) Write out the Boolean expressions for the Sum (S) and Carry (C_{out}) outputs of a 1-bit FA only using ANDs and ORs (no XORs). Recall: Inputs are A, B, and C_{in}. Assume both true and compliments of all inputs are available.

b) (5 points) Draw the logic circuit for the 1-bit FA only using 2-input NANDs, NORs, and inverters. Please do not use XOR or XNOR logic gates. Assume both true and compliments of all inputs are available.

One characteristic we have not discussed in class is propagation delay through logic gates, i.e., time delay from when inputs to the logic gate changes to when the output changes. Assume for the remaining questions that each logic gate has a propagation delay of 1 time unit, τ. Assume there is no delay for signals to propagate through wires.

c) (5 points) What are the delays to compute the outputs of a 1-bit FA (S and C_{out}) when the inputs (A, B, and C_{in}) change? Again, you can assume both true and compliments of all inputs are available.

d) (5 points) Let's now build an 8-bit adder by cascading eight 1-bit FAs together in a row where the C_{in} comes from the C_{out} of the lower-order bit's FA block. Draw the 8-bit adder circuit. You may abstract the 1-bit FA as a box with inputs and outputs (as seen in lecture notes).

e) (5 points) What you designed above is called a ripple-carry adder, because the carry ripples though the adder from the lowest order bit to the higher order bit. For your 8-bit ripple-carry, what is the longest amount of time it can take to compute the sum for an arbitrary set of 8-bit inputs A and B? How many units of τ does it take? Hint: Think about what happens when you add A = 1111_1111 and B = 0000_0001.

Problem 3: Flip-Flops (10 points)

You've seen in lecture that there are different kinds of flip-flops. Sometimes we don't have every single kind of flip-flop readily available, but we can use some latches and flip-flops to make other flip-flops. For example, we can make a D flip-flop or a J-K flip-flop by adding circuits to S-R latches.

We can create a toggle flip-flop (T flip-flop) by adding logic gates to a D flip-flop. The output of the T flip-flop's output (Q) toggles (0→1 or 1→0) whenever the input (T) is high on each rising edge of the clock (clk) signal.

a) (5 points) Write out the truth table for the T flip-flop.

b) (5 points) Draw the circuit for the T flip-flop. Feel free to use the box version of the D flip-flop, i.e., no need to draw the logic gates for the D flip-flop.

Problem 4: Image Compression (10 points)

How much space (in bytes) would the following images take up on your hard drive?

a) (2 points) 320x240 black and white image

b) (2 points) 320x240 gray-scale image

c) (2 points) 320x240 true color (RGB) image

d) (2 points) 320x240 indexed image with 256 colors

e) (2 points) What is the compression ratio between (c) and (d)?

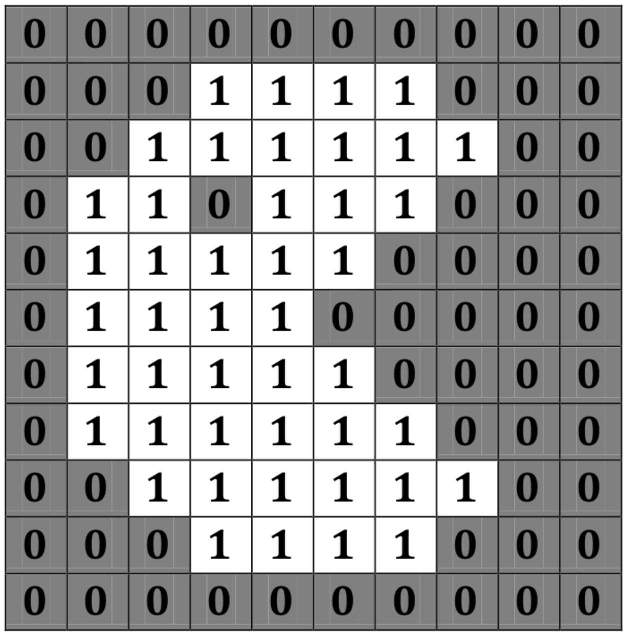
f) (2 points) What are the advantages and disadvantages of using indexed colors? For what type of image would we use indexed colors?

Problem 5: Image Processing (16 points)

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	1	1	0	1	1	1	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

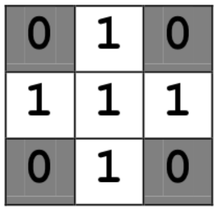
a) (8 points) Perform a closing on the above image using the following structuring element.

1	1	1
1	1	1
1	1	1



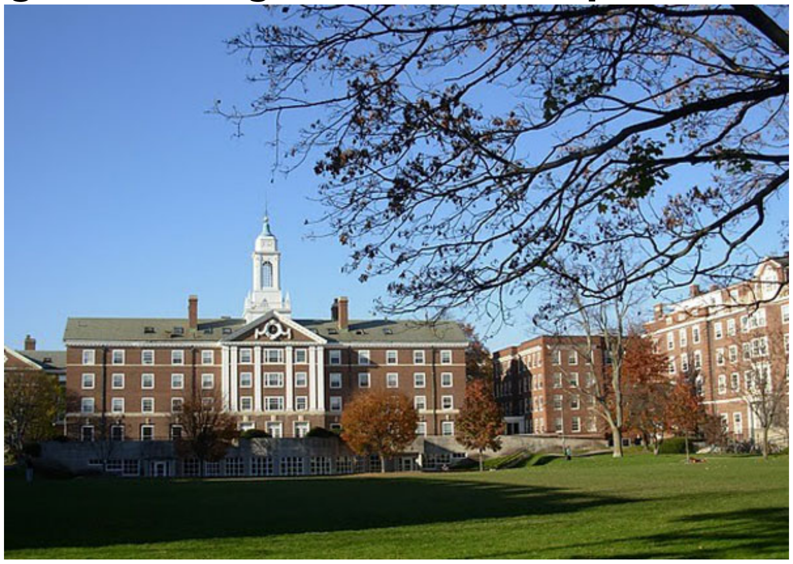
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	1	1	0	1	1	1	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

b) (8 points) Perform a closing on the above image using the following structuring element.



0	1	0
1	1	1
0	1	0

Problem 6: Image Processing with Matlab (26 points)



Download the image quad.jpg (above) and save it to whichever folder you use for Matlab work. Submit all code used for parts (b) – (g) below. Evidence that the code works (e.g., for pictures) can be submitted through screenshots. Please print out your solutions and turn them in. An easy way to do this is to use the "Publish" feature in MATLAB. However, make sure to go to the Publish options and publish your document as a PDF. Printed solutions can have color pictures in black and white or grayscale.

*Please make PLENTIFUL use of the "%" to comment on your code and let us know what you were trying to do. Or add a semi-detailed overview before the code. This way, we can give you partial credit.

a) (2 points) What is the height and width of the picture?

b) (4 points) Write a code to show a grayscale version of the picture. Display the color image and the grayscale version side by side, with the color image on the right and grayscale on the left.

c) (6 points) Show the red, blue and green channels of the image in a single figure using subplot command.

d) (2 points) Show 3 pictures in a horizontal strip, and one below the other in a vertical strip (manipulate the subplot command).

e) (2 points) Similarly, show the histograms for each of the color channels (in a subplot).

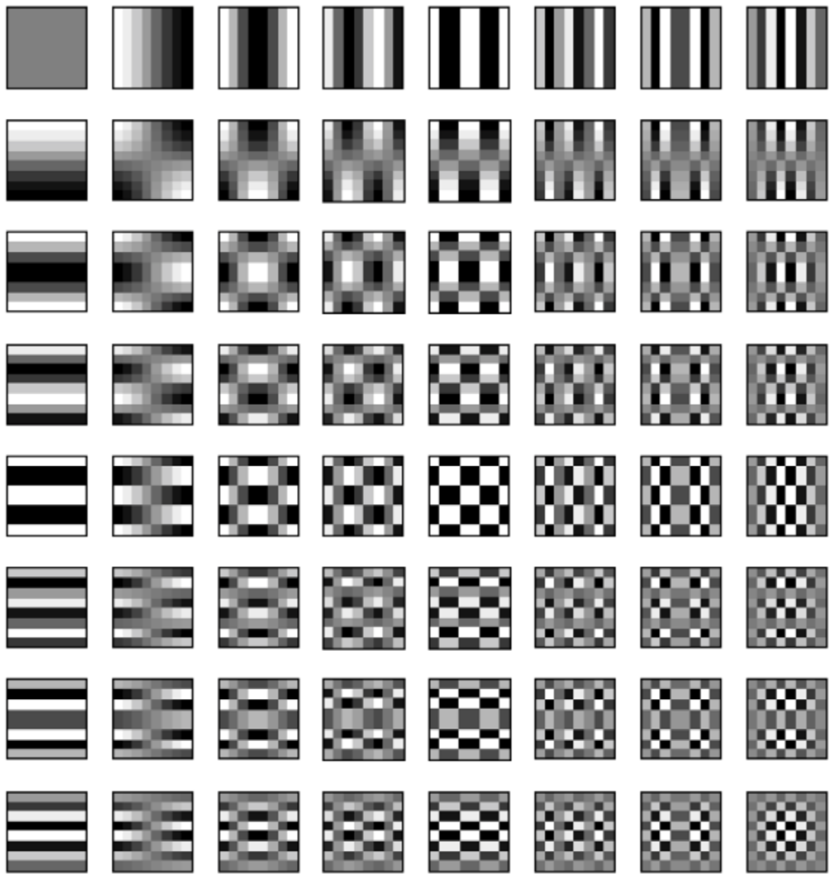
f) (5 points) Use the information you have to separate the sky from the rest of the image (your final image can be black and white).

g) (5 points) Now separate something else in the image, e.g., grass, buildings, etc.

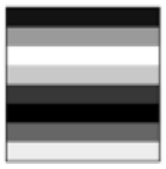
h) Do you give us permission to share your solution with the class if the staff finds it creative/interesting?

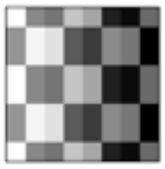
Bonus: Discrete Cosine Transform (12 points)

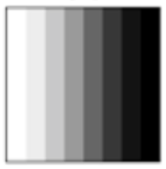
Images can be written as a weighted sum of the 64 basis images in the figure below.

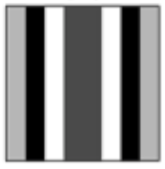


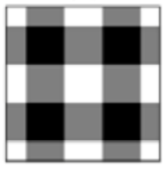
a) (12 points) Match each of the 8x8 images below on the left to its DCT on the right.

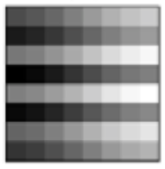

(a)


(b)


(c)


(d)


(e)


(f)

$$\begin{bmatrix} 100 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(i)

$$\begin{bmatrix} 100 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(ii)

$$\begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(iii)

$$\begin{bmatrix} 100 & -10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(iv)

$$\begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(v)

$$\begin{bmatrix} 100 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
(vi)