CS100M Fall 2007

Grading Guide: Project 4

March 31, 2008

The coded items below (e.g., c1e, s2a) indicate what a student's solution should accomplish. Codes that begin with the letter "c" deals with correctness; codes that begin with "s" deals with style.

Grader

If a student's solution does not accomplish task c1a, for example, then write the task code "c1a" along with any diagnostic remarks you can give. Count the number of correctness and style errors separately.

Items marked with ** count as two errors. In the table below, the top row lists the possible scores (1 to 5). The next row lists the number of correctness errors corresponding to every score category. The style score is determined similarly. Enter the total score (maximum of 10) in CMS as the project score. If there are bonus questions, enter any bonus points separately in the "Bonus Bucket", separate from the project score.

Student

Read the grading guide for every project, even if you get a perfect score! Notice from the table below that we often give one or two "freebies", i.e., mistakes that don't cost you any points. Learn from working on the project, and learn from any mistakes.

Scores

- c ands stand for correctness and style; see table below.
- parts with ** next to them means that they are double the value, *** for triple, etc.

Score	5	4	3	2	1	0
#correctness errors	$0-1$	$2-3$	$4-6$	$7-9$	$10-15$	$\geqslant 16$
#style errors	$0-2$	$3-5$	$6-8$	$9-12$	$13-19$	$\geqslant 20$

General

- (s0a) Use meaningful variable names
- (s0b) Appropriate indentation
- (s0c) Appropriate and concise comments throughout.
- (s0d) [up to 6*, one for each m-file] Comment header is *required*.
- (s0e) Reasonable line lengths; no horizontal scrolling
- (s0f) [up to **] No superfluous code
- (s0g) [up to **] Reasonably efficient and concise code; a little inefficient is OK
- (s0h) No debugging output
- (c0a) [2* max]Program successfully executes without crashing.(*for occasional,**for persistent)

1 Image Manipulation Tools

1.1 Swap

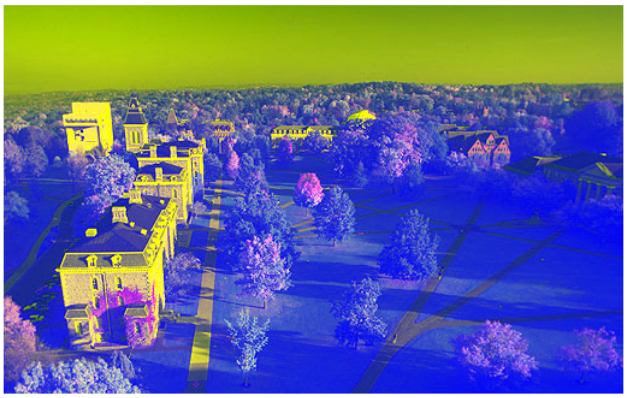
- c1a : Correctly defines size of output image, copies other elements correctly.
- c1b : Swaps over the defined components.
- s1a : Image is read in as an array (not filename), Components are specified as integers.
- c1c : *imshow(swap(imread('small.jpg'),1,2));* displays correctly.

1.2 Scale

- c2a : Correctly defines size of output image, copies other elements correctly.
- c2b : multiplies each component correctly.
- s2a : Image is read in as an array (not filename), Components are specified as integers.
- c2c : *imshow(scale(imread('small.jpg'),1, 0.5, 0.8));* displays correctly.

1.3 Invert

- c3a : Correctly defines size of output image, copies other elements correctly
- c3b : Inverts the component correctly.
- s3a : Image is read in as an array (not filename), component is specified as an integer.
- c3c : *imshow(invert(imread('small.jpg'),3));* displays correctly.



1.4 Modify

- c4a : Correctly defines size of output image, copies other elements correctly.
- c4b : Calls the function correctly on the specified component.
- s4a : Image is read in as an array (not filename), Component is specified as an integer.
- c4c : *imshow(modify(imread('small.jpg'),3, @mirror2));* displays correctly.

2 Hiding an Image

2.1 Hide

- c5a : Correctly defines size of output image.
- c5b : Obtains the correct size of the B&W image (no need to check that it is $<$ color image)
- c5c : Correctly iterates over the x and y components (while/nested for).
- c5d : Splits the 8 bit B&W value into 3 2-bit R, G, B components (last 2 bits are discarded). Note: cannot use dec2bin, bin2dec
- c5e : Encodes the change correctly, checks boundary conditions ($<$ 0 or $>$ 255)
- s5a : Does all updates in one pass (inefficient to do one component at a time)
- s5b : Image is read in as an array (not filename).
- c5f : *imshow(recover(imread('small.jpg'),hide(imread('bworiginal.jpg'),imread('small.jpg'))))* works. Should display the B&W image, seen below.

2.2 Recover

- c6a : Correctly obtains the size of the input data.
- c6b : Conservatively assigns size of B&W image (i.e. input size at max)
- c6c : Correctly iterates over the x and y components correctly (while/nested for).
- c6d : Combines the R, G, B components into a B&W value correctly (2 trailing zeros). Note: cannot use dec2bin, bin2dec
- c6e : Handles boundary cases correctly (checks for addition vs. subtraction: important in numbers near 0 or 255)
- s6a : Does all updates in one pass, inefficient to do one component at a time.
- s6b : Image is read in as an array (not filename), correctly returns the B&W image.
- c6f : *imshow(recover(imread('small.jpg'), imread('hidden.bmp')));* displays correctly in the top left corner.