# CS 1301 <br> Individual Homework 7 - Ascii Art Due: Monday April $4^{\text {th }}$, before 11:55pm Out of $\mathbf{1 0 0}$ points 

Files to submit: 1. HW7.py<br>THIS IS AN INDIVIDUAL ASSIGNMENT!

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

## For Help:

- TA Helpdesk - Schedule posted on class website.
- Email TA's or use Piazza Forums


## Notes:

- Don't forget to include the required collaboration statement (outlined on the syllabus).
- Do not wait until the last minute to do this assignment in case you run into problems.
- Read the entire specifications document before starting this assignment.


## Ascii Art

For this homework, you will be creating a series of functions to help you create ASCII representations of images. ASCII is a standard that defines numbers to represent characters (remember that computers only deal with numbers). For example, under ASCII the number 65 (1000001 in binary) represents the character "A". You don't need to worry about the gritty details of ASCII; just know that we are taking an image and converting it into characters that represent that image.


Your ultimate goal is to write the final function makeAsciiImage(). But to make that happen, you'll need some helper functions first. We recommend that you write and test these functions in the order presented, as the earlier functions are useful for writing the later functions.

## Function name: makeGrayscale

parameters: a picture
returns: a picture
The first function you should write is makeGrayscale(), which takes in a picture as a parameter and returns the same picture in shades of gray (gray scale). Remember that a pixel is gray when each of its RGB values is the same. Hint: each color should have the same relative weight to the others, for example the three different colors $(60,0,0),(0,60,0),(0,0,60)$ would all be converted to the RGB value of $(20,20,20)$ by the makeGrayscale() function.

Remember to return a changed picture from this function.
show (makeGrayscale(somePic))
should show somePic in gray scale. Also, DO NOT modify the original picture, so the somePic in the previous example should NOT change. You will need to create a new image to return within your function (possibly by copying the input picture.)

## Function name: getBlockValue

parameters: a picture, an x coordinate, a y coordinate
returns: an integer between 0-255 inclusive
When you convert an image to its ASCII representation, you are using ASCII characters as if they were pixels. An issue with this is that pixels are square, but characters are rectangular. You can solve this problem by representing blocks of pixels with characters instead of just one pixel. For the purpose of this project, a block is defined as 2 pixels wide and 4 pixels tall. The following text shows 6 blocks of pixels (pixels with the same number are in the same block).

You need to write a function that takes in a picture, an x coordinate, and a y coordinate, and returns the average value of the pixels contained in the block. The $x$ and $y$ coordinate represent the upper left of the pixel block. So if I called getBlockValue(somePic, 0,0 ), I would get the average intensity value of the pixels at $(0,0),(0,1),(0,2),(0,3)$ and $(1,0),(1,1),(1,2),(1,3)$. Hint: since we have already written makeGrayscale, you are expecting the pixels to be gray scale. You can use this to simplify the process of getting the average intensity value of every pixel in the block.

## Function name: getASCIIPixel

paramters: a value between 0-255 inclusive
returns: a character
Now that you have a way of determining how light or dark a pixel block is, you need a way of determining what character should represent that block. For this assignment, you should use this 10 value scale $\rightarrow$

| $0-25.999$ | $26-50.99$ | $51-75.99$ | $76-100.9$ | $101-$ <br> 125.9 | $126-$ <br> 150.9 | $151-$ <br> 175.999 | $176-$ <br> 200.999 | $201-$ <br> 225.999 | $226-255$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| $@$ | $\#$ | $\%$ | $*$ | $=$ | + | - | $:$ | . |  |
| At | Hash | Percent | Asterisk | Equal | Plus | Hyphen | Colon | Period | Space |

You could represent this as a string: "@\#\%*=+-: . "
with the the "@" representing the darkest color and a " " (space) representing the lightest.
The next function you should write is getASCIIPixel, which takes in an average RGB value as a parameter and returns a character that represents that average value. Use the knowledge that the passed in value will only be between $0-255$ and that there are 10 total possible values to determine which character to return. (Hint: remember that you can get a specific character out of a String by using bracket notation)
getAsciiPixel(0) returns "@"
getAsciiPixel(255) return " " (a space character)

## Function name: getASCIIMatrix

parameters: a picture
returns: a two dimensional list ([ [~], [~], [~] ,..... ])

Now it's time to bring everything together. Write the function getASCIIMatrix that takes in a picture, and outputs a two dimensional list containing the ASCII pixels that represents the inputed picture.

## Before you start writing this, plan how you will use all of the functions you have written to accomplish your goal. If you make a plan, implementing this function will be much easier.

You will probably want to use nested for loops when you are iterating through all of the pixels in the picture. You will need to set up these for loops in such a way as to use your getBlockValue function with no overlap (you never check the same pixel more than once; you should also make sure to check every pixel at least once). Assume the picture you are passing in is NOT already black and white. The output is a list, with other lists inside of it. A few examples follow to clarify how you should format the output.

If this is my picture:
$==$
$+++$
my output would look like this: [["=","=","="], ["+","+","+"], ["-","-","--"]]

## Function name: makeASCIIArt

parameters: a picture name (like something.jpg)
returns: None
saves to the file: a text file with your ASCII art (something.txt)
This is the final function you will write. makeAsciiArt must take in the name (of the picture we want to convert into ASCII (make sure it's in the same directory as this python file). The function will not return any value, however it must save a new text file with your ASCII art with the same name as the picture (If I input "cat.jpg", the text file should be named cat.txt). Take note that you will need to create a picture using the supplied picture file path. After you are done writing to your text file, make sure to close it!

## FINAL NOTES

We have provided you with two pictures to test with. Make sure to put them in the same directory as your python script.

Make sure to test your functions one at a time throughout this homework. This will make debugging MUCH easier. If you write all of your functions then run your script and it breaks, you won't know where to start fixing it.

## Rubric:

makeGrayscale
Takes in a picture and returns the same picture in black and white: 10 points

## getBlockValue

Returns a value between 0-255 inclusive:
Correctly reads values from all 8 individual pixels:
Returns the correct value:
getASCIIPixel
Generally returns the correct character from the string: 10 points Handles edge cases correctly:

5 points

## getASCIIMatrix

Returns nested lists: 5 points
Returns nested lists in the correct format ([[row0],[row1] ....): 10 points returned lists have the correct values ([[pixel0, pixel1, ...], ..] : 15 points
makeASCIIArt
5 points
5 points
5 points

## 100 points total

10 pts

15 pts

15 pts

30 pts

Creates a text file in the same directory with proper name:
That contains an ASCII representation of the inputed image:
10 points
20 points

