http://cs.ucf.edu/~bagci/

[Programming Assignment] (0/BONUS)

Computer Vision

Dr. Ulas Bagci • (Fall) 2016 • University of Central Florida (UCF)

Coding Standard and General Requirements

Code for all programming assignments should be **well documented**. A working program with no comments will receive **only partial credit**. Documentation entails writing a description of each function/method, class/structure, as well as comments throughout the code to explain the program flow. Programming language for the assignment is **Python**. You can use standard python built-in IDLE, or CANOPY for the working environment. Other commonly used IDLEs are the following: PyCharm Community Edition, PyScripter, CodeSculptor, Eric Python, Eclipse plus PyDev.

Following libraries will be used extensively throughout the course:

- PIL (The Python Imaging Library), Matplotlib, NumPy, SciPy, LibSVM, OpenCV, VLFeat, python-graph.

If you use CANOPY, make sure that you use version 2.7, which already includes many libraries. If you are asked to implement "Gaussian Filtering", you are not allowed to use a Gaussian function from a known library, you need to implement it from scratch.

Submit by **3rd of September 2016**, 11.59pm.

This assignment was prepared to motivate you to program with Python and get hands-on experience in numerical problems.

Problem 1

[0.5 pt] A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward. For instance, 121, 90011211009, 34243, 33, all are palindrome numbers. Four different numbers between 10 and 1000 are Palindromes, **their summation is also Palindrome**. Write a Python script that finds the maximum value of multiplication of these 4 **distinct** palindrome numbers between 10 and 1000. Results as well as these four numbers should be written in the screen.

Problem 2

[0.5 pt] Ask user to select an upper bound (X) for an interval: $(1, X)$, and then ask user again to pick an integer number between $(1, X)$ in their mind. Next, your script will estimate what you have in your mind, and when program's estimation is larger than what you have in your mind, you need to input "H", when program's estimation is smaller than what you have in your mind, you need to input "L". If program estimate your number, then you need to type "W". Your program then prints number of attempts, and your number in the screen. Please note that, you need to develop an algorithm which optimizes this game with the smallest number of attempts.

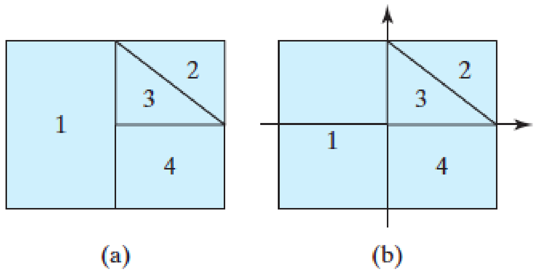
Problem 3

[0.5 pt] (a) A positive integer is called a **perfect number** if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because $6 = 1+2+3$. The next is $28 = 14+7+4+2+1$. There are four perfect numbers less than 10,000. Write a program to find these four numbers.

[0.5 pt] (b) Generate 9 numbers between 0 and 1 using *rand*, and create 3 by 3 matrix. Next, assess whether matrix is singular or not. If not, calculate determinant of it. Write every piece yourself except generating numbers.

Problem 4

[1 pts], (Monte Carlo Simulation) A square is divided into four smaller regions as shown in (a). If you throw a dart into the square one million times, what is the probability for the dart to fall into an odd-numbered region? Write a program to simulate the process and display the result. (**Hint:** Place the center of the square in the center of a coordinate system, as shown in (b). Randomly generate a point in the square and count the number of times for a point to fall in an odd-numbered region.)



Problem 5

[1 pts], (Simulation program, Use Matplotlib) Imagine three ants sitting at the vertices of an equilateral triangle with side lengths of 1 mile. At the same instant, each ant begins walking and continues walking at s miles per hour directly at the ant to its right. When do the ants collide?, how far do they walk?, and along what path do they walk? Please give separate answers to each question, and draw path of each ant in the same graph/triangle.

[**Hint:** the goal is to write a program that tracks the movement of each ant, and we have to follow all three ants because the motion of one ant determines the motion of the other two ants. The ants move continuously along smooth curves as they approach each other. However, a computer cannot store an infinite number of points, so it cannot exactly reproduce truly continuous motion. Instead, we imagine the ants moving in a sequence of many small steps. To do this, we use a time step Δt that can be chosen and varied by the user. At each time step, each ant finds the direction that it should walk (so that it walks directly toward its neighbor) and then takes a step of length $L = s\Delta t$ in that direction. As the ants move, they continue to be at the vertices of an equilateral triangle that is rotating and shrinking. So the paths of the ants differ only by rotations of $(2/3)\pi$ radians.]

Problem 6

[0.5 pts], (Persistence difference) Pick a four digit number (your program asks users to enter), from its digits, form the smallest and largest four digit numbers (m and M, respectively). Find (M-m). Keep repeating the procedure, until you reach a number which does not change anymore. What is this number (show on the screen soon after user enters its input)? What numbers will be reached if two, three, and five-digits numbers are used for this procedure?

[**Hint:** Treat 397, let say, as 0397 in case you have 0 in the digits of the number.]

Problem 7

[0.5 pts], (Matrix Manipulation) Create a blank matrix with a size of 200 by 200. For each element of the matrix (i.e., pixel), randomly assign an integer value of either 255 or 0 (i.e., probability of a pixel having 0 or 255 as an intensity value is 0.5). Now find all the pixels in the matrix with non-zero values, and for each of them, check their 4-neighbors (east, west, north, and south). If all neighbors are having zero values, then keep the pixel's originally assigned value (which is 255); otherwise, change the intensity value by half (i.e., 128). Continue this process until the matrix does not change anymore. Show the final matrix on the screen as an image with grayscale.