ISC-3313 Final Project

Instructor: Eitan Lees

Summer 2018

In order to satisfy FSU's computer competency requirement, each student must complete a computer project. This project represents 40% of the final grade for the course. The project involves:

- a short written report describing the project and the program (around 3-5 pages would be fine);
- a working python program, or jupyter notebook, which you have written;
- an oral presentation (about 5 minutes);

Oral presentations are scheduled for **July 31st**, the second to last day of class. You may make your presentation in a number of ways:

- just talking;
- using the whiteboard;
- using Powerpoint slides or other programs to display your talk;
- going to the computer if you need to run your program;

Your presentation will be short. You should be sure to include the following important information:

- a simple explanation of the problem you are working on;
- a small example of your problem;
- a description of the algorithm (the ideas) that you are using to solve the problem;
- how you used python to define your problem, determine the solution, and output the results.

Your report and program are due **August 2nd**, the last day class. You should submit them to canvas.

I will NOT be grading your report on proper English, or spelling. I DO expect that your report will contain the information, in written form, that I asked you to present in your oral report:

- a simple explanation of the problem you are working on;

- a small example of your problem;
- a description of the algorithm (the ideas) that you are using to solve the problem;
- how you used python to define your problem, determine the solution, and output the results.

For labs and homeworks, I have NOT required you to comment your programs. But, since this program represents your computer competency project, I expect you to try to make the program neat and readable. I expect your program to have some comments or explanations that tell me what the important variables are, and how the program is solving the problem. You do not have to explain every single statement in your program, but you should point out the parts that make the computation work.

The following is a list of suggestions for projects. In cases where a reference is listed, I might be able to get you a copy to look over. Instead of choosing a project from this list, you may instead think up one on your own, and propose it to me.

If you are interested in a project, please let me know and we can discuss it. E-mail me (elees@fsu.edu) on or before **June 19th** with a 200 word outline for your project.

Project List

- **Card Counting**: simulate the game of blackjack. (First you need to find the rules. Assume, for instance, that both you and the dealer always keep drawing until you reach 17 or more.) Be sure to use a single deck of cards, and play until you run out of cards. Bet $10 on every hand, and keep track of your winnings at the end of each deck. Now find out the basic card counting rules: for a given deck, count every face card and 10 that has been played as -1, and all other cards as +1. Whenever the count is negative, bet $1; if it is positive, bet 10 times the count. Do you see any benefit to a strategy like this? Reference: `http://en.wikipedia.org/wiki/Card_counting`
- **The Carefree Wanderer**: an important model in physics is called the random walk. In a computational model, we start at the origin, choose a direction (perhaps north/south/east/west) at random, and take a step. We repeat this process n times, and ask how far we are likely to be from our starting point. References: Martin Gardner, "Random Walks on the Plane and in Space", in The Mathematical Circus.
- **The Careful Wanderer**: a variation of the random walk adds one condition: the walk can never cross itself. If we start at the origin, we choose at random from those directions we have never taken, and take a step, and do that n times. For this problem, it's possible to get boxed in, unable to take a step. So the first question is, if you are working in 2D, and want to take n "careful" steps, what are the chances you can actually complete your task without getting boxed in? And if you make it, are you typically further away from the origin than the careless drunk is? References: Brian Hayes, "How to avoid yourself", American Scientist, Volume 86, Number 4, July-August 1998, pages 314-319.
- **The Cellular Automaton**: a cellular automaton is a grid of cells, which can be on or off (0/1), and a set of rules for how grid cells change based on the state of their

neighbors. A 1 dimensional cellular automaton is simply an array of n values. On each step, all cells check their left and right neighbors, and then decide whether to turn on or off. Some cellular automatons are almost like random number generators, going through a complicated pattern of changes. References: Brian Hayes, "Computer Recreations: The cellular automaton offers a model of the world and a world unto itself", Scientific American, March 1984, Volume 150, pages 12-21.

- **The Cheapest Rug**: your new house has beautiful floors, except for one area where there are n random holes. What's the size of the smallest circular rug you can buy that will cover all the holes? This problem is easy for three holes. But if you can solve that problem, you're halfway to a solution of the general problem. References: de Berg, van Kreveld, Overmars, Schwarzkopf, Computational Geometry, Section 4.7, "Smallest Enclosing Discs", Springer, 2000.

- **The Damaged Picture**: suppose we have a beautiful picture which has been damaged by small imperfections, perhaps small black and white particles have fallen all over it. Even though we can sort of still see the picture, it's unpleasant to look at. Can we restore the original image, or at least come close? Write a C program that reads in a damaged digital image and uses the median filter to try to clean it up. Reference:`http:en.wikipedia.org/wiki/Median_filter`

- **The Day Calculator**: how many days old are you? How many days elapsed between the beginning and end of the American Civil War? If you know the calendar dates of two events, you should be able to count the number of days between them, but it's not a trivial task. However, with a little bit of thought, you can work out the formulas you would need to do this. One idea is called the "Julian Day Number", which assigns a unique counting number to every day in history. References: Lance Latham, Standard C Date/Time Library, RD Books, 1998.

- **The Factorial Game**: We think of the climate as the average of the weather, and whenever we are faced with a random process, we try to understand it by asking what it would be on average. But here's an example to show that we can't always do this. Let's define a game in which the player picks a number n, and then is given a die with the numbers 1 through n on it. The player rolls the die over and over again, recording the result, until rolling a 1. The player's winnings are the product of all the numbers rolled. To play the game once costs n! dollars, where n! is the factorial of n, that is, the product of the integers from 1 to n. Is this a fair price? To answer that question, we could simulate playing the game 100 times, and compare the average winnings to the cost. But if we repeat the calculation for 1,000 plays, the average winnings change...a lot! Trying to get the average to settle down by taking a bigger and bigger sample seems to make things worse. This is an example of a random distribution for which we cannot compute an average value! References: Brian Hayes, "Fat Tails", American Scientist, Volume 95, Number 2, March-April 2007, pages 2000-2004.

- **Flip Sorting**: sorting a list of numbers is an important task in computing. Suppose you were asked to sort such a list, but the only way you are allowed to modify the list is to reverse the order of the entries in a consecutive part of the list. You can do that as many times as you want to. Can you sort the list? How fast can you sort it? What is the relationship between this operation and genetic variation in animals?

References: Brian Hayes, "Sorting out the genome", American Scientist, Volume 95, Number 5, September-October 2007, pages 386-391.

- **The Gambler**: a man knows that he must pay a debt of $20,000 on Monday or he is ruined. He only has $10,000. He goes to Las Vegas for the weekend and proceeds to gamble. If he bets $10 at a time, and the odds are even, then what are the chances he will reach his goal? On average, how many bets will be made before he wins $10,000 or loses all his money? References: Martin Gardner, "Random Walks and Gambling", in The Mathematical Circus.

- **The Mortgage Mess**: When you buy a car or a house, you generally take out a loan, and then make regular monthly payments. The amount you owe increases based on a given interest rate, and decreases because of your payments. What this means is that your first payment of $1,000 might only decrease the amount you owe (the principal) by $10 - the rest goes to paying interest. Towards the end of your payments, the situation is reversed. Given a loan amount, a loan time, and an interest rate, can you determine the size of the monthly payments? Can you print a table showing how much the principal has decreased each month? Can you show how much faster you would pay off the loan if you increased your monthly payments by $10? References: Dwight Neuenschwander, Elegant connections in physics: Thinking like a physicist about amortization schedules, Radiations, Volume 16, Number 2, Fall 2010, pages 19-23.

- **The Partition Problem**: Suppose an art collector has died, and the will requires that the collection be divided up, as equally as possible, between two heirs. We can model this problem by assuming we have a list of n integers, and that we wish to split the list into two parts whose sums are equal, or as close as possible. Can you work out a program that tries to solve this problem? When n is large, a perfect answer might be hard to find; can you think of a simple approach that might come close to solving the problem? References: Alexander Dewdney, "The Partition Problem" in The Turing Omnibus; Brian Hayes, "The Easiest Hard Problem", American Scientist, Volume 90, Number 2, March-April 2002, pages 113-117.

- **Quasirandom Numbers**: Computers can generate random numbers. If these were darts, you would see them scattered across the dartboard, with some big gaps here and there. The gaps and clusters in random numbers are a problem if we what we really want to do is to explore typical values of some quantity. A special approach, called "quasirandom" numbers, can produce better samples than the typical random number approach. In this project, I'd like you to investigate how random numbers can be used to estimate integrals or areas, and then compare your results with a quasirandom method. Reference: Brian Hayes, Quasirandom Ramblings, American Scientist, Volume 99, Number 4, July-August 2011, pages 282-287.

- **Random Numbers**: It is somewhat surprising that a computer needs to determine random numbers, and even more surprising that it can...almost. There are many formulas for determining a sequence of "pseudorandom" numbers. In this project, I would like you to look at what are called LCRG's, or "linear congruential random generators". These simply use a starting value x0 and repeatedly apply a formula like x1=a*x0+b (along with only keeping the fractional part of the result). I will help you choose a couple LCRG's to investigate. Your job is to write a program that

can generate these sequences, find out if the sequence repeats, and how soon, and look at whether the random numbers generates by such a process produce results that really do seem random. Reference: Stephen Park, Keith Miller, Random Number Generators: Good Ones are Hard to Find, Communications of the ACM, Volume 31, Number 10, October 1988, pages 1192-1201. Brian Hayes, "The wheel of fortune", American Scientist, Volume 81, Number 2, March/April 1993, pages 114-118.

- **The Rectangle Challenge**: Could you make a recognizable picture of yourself using just 32 pixels? What if I allowed you to use 32 rectangles, which could be of different sizes and colors. The rectangles can overlap, and if they do, their colors are averaged over that part. This is a problem with two important features: we have no idea of how to find a solution, and we have a way of evaluating any attempted solution (we'll compare it to a given photograph of you.) Problems like this can sometimes be handled by "genetic algorithms". A genetic algorithm takes a group of bad solutions, discards the worst, and then essentially "breeds" a new generation of solutions by mixing up the information in the current solutions. References: Nick Berry, "A practical use for genetic programming", `http://www.datagenetics.com/blog.html`

- **The Solar System**: put together formulas describing the locations of the planets over time. Given a date, print out the corresponding locations. References: Tanmay Singal and Ashok Singal, "Determining planetary positions in the sky for +/-50 years to an accuracy of 1 degree with a calculator", PRAYAS Students Journal of Physics, `http://arxiv.org/abs/0910.2778`.

- **The Solitary Cards**: Suppose we have a deck of 52 cards, and we place them randomly on a table. Can you identify those cards which are not touching any other cards? To make this doable on a computer, assume each card is 5 units wide and 7 units high, that the table is 500 units wide and 500 units high, and that each card stays in its original orientation, not being turned at an angle. Moreover, assume that the lower left corner of the card is always an integer coordinate between (0,0) and (495,493). You should be able to figure out how to place the cards randomly on the table. You should be able to figure out which cards are not touched by any other card. I can help you create a plot at the end, showing the positions of the cards, with the "solitary" cards highlighted.

- **Statistics of Money**: when the distribution of wealth in a society is described by a histogram, many societies seem to follow a common pattern. Simulate an economy to study this idea. Each person starts with the same amount of money, and then random pairs of people meet and bet against each other. The most that can be bet is the amount that the poorer person has (randomly choose an amount between$0 and this upper limit.) A random coin toss decides who gets the money. Repeat this process and see what happens. A modification to the betting limit gives you a different result, that is more like the kinetic energy distribution in a gas. References: Brian Hayes, "Follow the money", American Scientist, Volume 90, Number 5, September-October 2002, pages 400-405.

- **The Subset Sum Problem**: At a carnival, you see a game in which you can throw baseballs at small dummies. Each dummy has a number on it, and your task is to knock down only those dummies whose numbers will add up to 50. The dummies are numbered 25, 27, 3, 12, 6, 15, 9, 30, 21 and 19. Can you do it? How hard is this

problem in general? Are there any reasonable ways to look for solutions? References: Cormen, Leiserson, Rivest and Stein, "Introduction to Algorithms"; Berman and Paul, "Algorithms: Sequential, Parallel, and Distributed"; David Pisinger, "Linear Time Algorithms for Knapsack Problems with Bounded Weights", Journal of Algorithms, Volume 33, Number 1, October 1999, pages 1–14.

- **The Traveling Salesman Problem**: a traveler must plan a trip that visits each city on a list exactly once. The distances between cities are known (but in some cases there might be no direct link from one to another), and the traveler wants to minimize the total distance traveled. For 5 cities, this is easy; for 10 it becomes hard and quickly impossible. Nonetheless, there are ways to guess a good approximation, and ways to try to make an good travel plan even better. Get the list of locations for the 48 US state capitals from me, and try some of these methods! References: Charles Van Loan, Daisy Fan, "Insight Through Computing", SIAM, 2010; `http://www.tsp.gatech.edu/`, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`, David Cook, "In Pursuit of the Traveling Salesman", ebook available through FSU library.

- **Waiting Times**: a bank has T tellers who are available to serve customers. Customers arrive at random times, and have random amounts of business to transact. When the bank closes at 5pm, can we determine the average time a customer had to wait in a queue before being served? We need to be able to simulate a typical bank day, over and over, and then average the results. This requires that we be able to tell the computer how to create a customer at a random time, and give the customer a random amount of work, and add the customer to the queue. References: A K Dewdney, Simulation: The Monte Carlo Method, in The Turing Omnibus.