Guidelines for the Final Projects
Introduction to Scientific Computing with C++
05 July 2011

---

1. This course requires a final “capstone project” in order to fulfill FSU’s Computer Competency Requirement. The student, with the guidance and approval of the instructor, selects a computer project to work on. Completion of the project requires

1. a working C++ program;
2. a written report (3-5 pages);
3. an oral presentation (5 minutes).

2. Each student should submit a one-paragraph project proposal, written, printed, or by email, by class time on Tuesday, July 12th. The proposal should describe the topic to be explored, discuss the interesting features of the project, and outline the program that is to be written and what it will do. Students are free to propose a project of their own choosing, but a list of suggested topic is provided below, and students unable to decide on a project topic will have one assigned.

3. The projects are due on Tuesday, August 2nd. At that time, the programs and reports must be turned in, and each student will make an oral presentation summarizing the project.

4. The written report should be between 3 to 5 pages long. It should describe the topic, explain any mathematical or scientific background of interest, discuss why the topic is of interest or use to the student, outline the program, and present and analyze the results.

5. The oral presentation should be about 5 minutes long. The presenter may use slides, or write on the board, or run a demonstration on the computer, but none of this is necessary. It is only required that the student be able to explain the project and the program in words.

Suggested Topics

Many of these topics come from articles by Brian Hayes in *American Scientist* or by A K Dewdney in *Scientific American.*

- **Flip-sorting**: how a list can be sorted when you are only allowed to reverse the order of a section of the list; Brian Hayes, *Sorting Out the Genome*;
- **A never-ending bet**: simulate a series of bets between two players. The loser loses half, and the bet is played again. What outcomes are likely? Brian Hayes, *Wagering with Zeno*;
- **The factoidal function**: similar to the factorial, but the factors are random. What is the average value of the function? Brian Hayes, *Fat Tails*;
- **Rumor spreading**: simulate the spread of a rumor through a population. Brian Hayes, *Rumours and Errours*;
- **The Vibonacci sequence**: like the Fibonacci numbers, except the two previous numbers are randomly added or subtracted to get the next one. Brian Hayes, *The Vibonacci Numbers*;
- **1D cellular automata**: the entries in an array are randomly initialized to 0 or 1. Then, a rule is applied to each entry, which changes it depending on its own value and its two neighbors. Each time the rule is applied, a new copy of the array is made. The changes in the array can seem random, or can exhibit strange patterns. Brian Hayes, *The Cellular Automaton*;
- **Quasi-random numbers**: quasi-random numbers are generated by a simple formula. They are more ordered than the random numbers we are used to, and less ordered than a grid. In some cases, they can provide a better area estimate than random numbers. Brian Hayes, *Quasirandom Ramblings*;
- **Simulated waiting time**: the arrival of customers in a bank, and time needed to carry out a transaction are simulated with a particular kind of random variable. The length of the customer line, and average waiting time, are of interest. A K Dewdney, *Simulation: The Monte Carlo Method*;

- **Even division of a list**: a method is described for determining whether a given list of numbers can be divided into two groups with the same sum. A K Dewdney, *The Partition Problem: A pseudo-fast algorithm*;

- **The game of life**: on a 2D checkerboard, some cells are 1, representing life. At each clock tick, a rule is applied which means some cells die, others survive, and some dead cells come to life. A K Dewdney, *Cellular Automata: the game of life*;

- **Minesweeper**: it is possible to set up a C++ program that allows the user to play a simple version of the minesweeper game. Instead of a graphical interface, the program prints out text that suggests the mine field. The user indicates the next choice by typing a row and column index; **http://en.wikipedia.org/wiki/Minesweeper_(video_game)**.

- **Calendars**: convert a date in from one calendar system to another. Convert a date in our current calendar to one in the ancient Egyptian calendar, for instance. E G Richards, chapter 25, *Mapping Time.*

- **Solar system**: find formulas describing the locations of the sun and planets over time. Given a date, print out the planetary locations. Tanmay Singal, *Determining planetary positions in the sky for* ± 50 *years to an accuracy of* 1° *with a calculator.*

- **Median filter for image processing**: given a 2D array of gray scale data describing an image, which contains some noisy pixels, try to clean up the image so that the noise disappears, using a simple median filter. The instructor can provide some noisy images, and suggestions for how to proceed. **http://en.wikipedia.org/wiki/Median_filter**

- **Gambler's ruin**: two players **A** and **B**, flip a coin repeatedly, staking $1 each time. They continue until one player is ruined, that is, has lost all money to the other player. If the players begin with **R** and **S** dollars respectively, then how long is the game likely to last? What are the odds that player **A** will win? Estimate these quantities by making 1,000 trials for each of a range of values **R** and **S**. **http://en.wikipedia.org/wiki/Gambler's_ruin**