PROGRAMMING PRACTICE SUGGESTIONS

Here are some suggestions for programming projects.

First, look at the template and introductory lecture to Mathematica which I've posted on my homepage; you can get these by going to
http://web.williams.edu/Mathematics/sjmiller/public_html/math/handouts/latex.htm

YouTube lecture: http://www.youtube.com/watch?v=g1oj7CIqGM8

Template:
http://web.williams.edu/Mathematics/sjmiller/public_html/math/LaTexMathematica/MathematicaIntroVer6.nb

Here are some practice programming challenges if you haven't programmed much.

Binomial coefficients:

1. Given a row n, write a program that prints out just row n of Pascal's triangle.

2. Redo above but now print out mod 2. Say do 0s and 1s, or blank space and *.

3. Redo above but now print out from rows n1 to n2 (you input those). DO NOT worry about making it pretty; you can do as just rows of text and don't worry on how it looks.

3x+1:

1. Write a program that takes a positive integer as input and returns 3x+1 if odd and x/2 if even.

2. Redo above but keep iterating until you reach 1. Print the path. Try 25, 27 and 29 as inputs.

3. Write a program to grab the leading digit of a non-zero positive real number, and record the number of each first digit as you iterate the 3x+1 map down to 1.

Counting Solutions:

1. Fix an a and an r, and consider x^2 + a y^2 <= r. Count how many solutions (x,y) in integers there are to this.

2. Redo the above when a=1 or a=4. Have the computer run for all r <= R and let R tend to infinity. Can you sniff out how the number of solutions grows with R? Can you sniff out an error term?

Horner's algorithm.

1. Read up on Horner's algorithm and write a program to use that to evaluate a polynomial (this will be assigned later this semester).

Binary expansions:

1. Write a program to return the binary expansion of a number. Note Mathematica has a function which does this!

2. Use the binary expansion to implement fast multiplication (will be assigned later this semester).

Monte Carlo Integration:

1. Choose positive functions f, g on [0,1] such that 0 <= f(x) <= g(x) <= 4. Randomly choose N points in [0,1] x [0,4] (so x-coord in [0,1] and y-coord in [0,4]). If the point (x,y) has f(x) <= y <= g(x) countit as a success, else count it as a failure. Consider #success/N as N becomes large; does this approach the area? If yes you've found a way to numerically integrate!

Random Walk:

1. Choose a probability p (say 1/2 to start), a positive integer L, and a starting value k. Start at k and flip a coin with probability p of heads. Every time you get a heads take one step to the right (so if you're at j go to j+1), else take a step to the left (go from j to j-1). Do this many times, and for different k record the probability you reach 0 before you reach L. This answer will depend on p, k and L; can you sniff out the relation?

Fibonacci numbers:

1. Write the Fibonacci numbers as 1, 2, 3, 5, 8, 13, ... so F_{n+1} = F_n + F_{n-1}. Find a formula to give F_n immediately (you can program Binet's formula, but you want to be exact and not a real number); Mathematica has a formula for the Fibonaccis preprogrammed.

2. Every positive integer can be written uniquely as a sum of non-adjacent FIbonacci numbers. This is called the Zeckendorf decomposition. You can find by using the greedy algorithm: given your number x subtract the largest Fibonacci you can, say F_i. Look at $x - F_i$ and subtract the largest Fibonacci you can. If that is F_{i-1} you could've subtracted $F_i + F_{i-1} = F_{i+1}$, contradicting the maximality of F_i. Return the Zeckendorf decomposition of x.

3. Building on the last project, create a histogram of differences between indices of summands in the Zeckendorf decomposition of large x. If $x = F_{19} + F_{11} + F_2$ the gaps are 8 and 9. Look at lots of x between F_n and F_{n+1} for n large.

4. Building on the last, for each x find the largest gap between indices of summands. Look at lots of x between F_n and F_{n+1} for n large. What do you see? Plot a histogram.

Perfect numbers:

1. A number n is perfect if the sum of its proper divisors equals itself. The first two are 6 and 28. Write a program to find all perfect numbers at most 1,000,000.

2. Fix a positive integer a. Say a number is a-perfect if the sum of its proper divisors is a less than itself. Find all a-perfect numbers at most 1,000,000.

Primes:

1. Write a program to factor a number (don't just use pre-existing functions).

2. Write a program to find all primes in a given range.

3. Given positive integers a and N, write a program to find N+1 primes $p_1, p_2, \ldots, p_{N+1}$ with $p_{i+1} - p_i = a$.

4. Write a program to determine all a at most 100 that are common outputs of the Euler totient function and the sum of proper divisors function. The Euler totient function phi(n) is the number of integers from 1 to n relatively prime to n; the sum of proper divisors function is the sum of the proper divisors of a number. If T is the set of all outputs of the totient function (so $T = \{a: a = \phi(n)$ for some $n\}$) and if S is the set of all outputs of the sum of proper divisor function (so $S = \{a: a = \mathrm{sum}_{d|n,\, d < n}\, d\}$), then we want to find S intersect T. Note that we don't need $a = \phi(n) = \mathrm{sum}_{d|n,\, d < n}\, d$; it's possible for $a = \phi(n)$ and $a = \mathrm{sum}_{p|m,\, p < m}\, d$.