Sept 12 Homework

1. Explain why the answer to Schumer problem 1.14 is F_{n+1}.

2. Twin primes are numbers n,n+2 that are both primes (eg: 11,13). Many large twin primes have been found. Triplet primes are numbers n,n+2,n+4 that are all three prime (eg: 3,5,7). Prove that there are no other triplet primes. (Hint: consider division by 3.)

3. Use your intuition to answer the following questions as best you can; there is no need to justify your answer, but there is a correct answer in each case.

   a. If $P(n) \equiv$[fraction of numbers between n-1000 and n+1000 which are prime], do you think P(n) tends to 0 or 1 or something between 0 and 1, as n tends to infinity?

   b. If N and M are randomly chosen large integers, the probability that N and M are relatively prime is about (multiple choice): 0%, 60% or 100%?

   c. Do you think mathematicians will ever discover an explicit formula $f(n)$ (like $n! + 1$ or $2^{2^n} + 1$, which often are prime, but not always) which generates a different prime number for every integer n?

4. Use Mathematical Induction to prove that $|sin(nx)| \leq n * sin(x)$ for any $0 \leq x \leq \pi$, for all positive integers n.

5. Prove that if two (diagonally) opposite corner squares are removed from a chess board, the remaining 62 squares cannot be covered by 31 normal (covering two squares each) dominoes.

6. The binary sort algorithm is defined recursively as follows: A list of $N = 2^m$ numbers is divided into two halves. The first $N/2 = 2^{m-1}$ numbers are sorted using the binary sort algorithm, and the last $N/2 = 2^{m-1}$ numbers are sorted using the binary sort algorithm. Then the two halves are merged, that is, the first number in the first list is compared with the first number in the second list, and the smaller becomes the first number in the final sorted list, etc. Naturally, if $N = 1(m = 0)$ there is nothing to do.

   The MATLAB program below implements the binary sort algorithm recursively.

```
      function F = bsort(F,M)
%                                IF M=0 (N=1), DO NOTHING
      if (M==0)
         return
      end
      N = 2^M;
      N2 = N/2;
%                                COPY FIRST HALF OF F ONTO F1
%                                AND SECOND HALF OF F ONTO F2
      F1(1:N2) = F(1:N2);
      F2(1:N2) = F(N2+1:N);
%                                SORT FIRST HALF OF F
      F1 = bsort(F1,M-1);
%                                SORT LAST HALF OF F
      F2 = bsort(F2,M-1);
%                                MERGE F1 AND F2 INTO F
      I1 = 1;
      I2 = 1;
      for I=1:N
         if (I1 > N2)
            F(I) = F2(I2);
            I2 = I2 + 1;
            continue
         end
         if (I2 > N2)
            F(I) = F1(I1);
            I1 = I1 + 1;
            continue
         end
         if (F1(I1) < F2(I2))
            F(I) = F1(I1);
            I1 = I1 + 1;
         else
            F(I) = F2(I2);
            I2 = I2 + 1;
         end
      end
```

If W_m is defined to be the number of comparisons required by the binary sort algorithm to sort 2^m numbers, use induction to prove that $W_m = m2^m$. Consider that the "merge" loop does N comparisons. (It actually does several other operations per loop pass, but only one comparison of numbers in our list.) Note that this means the binary sort algorithm requires only $Nlog(N)$ comparisons, MUCH faster than the usual "bubble sort" algorithm which requires about $\frac{1}{2}N^2$ comparisons.

7. If $E_0 = 2, E_{n+1} = (E_0 * E_1 * E_2 * ... * E_n) + 1$ (as in problem 4 of the last homework), either prove that all the E_n are prime, or find one which is not prime.