CSE 20, Discrete Mathematics, Spring 1997
Second Practice midterm
Answer Key

1. Prove that, for all $n \geq 1$, $\sum_{i=1}^{i=n} \frac{i}{2^i} = 2 - \frac{n+2}{2^n}$

   First, for $n = 1$, we have $\sum_{i=1}^{i=1} \frac{i}{2^i} = 1/2^1 = 1/2 = 2 - 3/2 = 2 - \frac{1+2}{2^1}$. So the equation is true when $n = 1$.

   Assume for some $k \geq 1$ that the equation is true for $n = k$, i.e., that $\sum_{i=1}^{i=k} \frac{i}{2^i} = 2 - \frac{k+2}{2^k}$ (Our goal is to prove the same thing for $n = k+1$, i.e. $\sum_{i=1}^{i=k+1} \frac{i}{2^i} = 2 - \frac{k+3}{2^{k+1}}$) Now, $\sum_{i=1}^{i=k+1} \frac{i}{2^i} = \sum_{i=1}^{i=k} \frac{i}{2^i} + \frac{k+1}{2^{k+1}} = 2 - \frac{k+2}{2^k} + \frac{k+1}{2^{k+1}} = 2 - \frac{2k+4}{2^{k+1}} + \frac{k+1}{2^{k+1}} = 2 - \frac{k+3}{2^{k+1}}$.

   Thus if the claim is true for $n = k$ then it is also true for $n = k+1$.

   Since it is true for $n = 1$, and if it holds for $n = k$ it is true for $n = k+1$ too, by induction, it is true for all $n \geq 1$.

2. I went to a party where some people shook hands. I shook hands with 5 people. Prove that there was someone else at the party who shook an odd number of people's hands. (Hint: Prove by induction on the number of hand shakes that the number of people who shook hands an odd number of times is even. In other words, for t the number of hand shakes, let Odd_t be the number of people who shook hands an odd number of times in the first t handshakes. Prove by induction that Odd_t is even. How can the number of "odd" people be changed by one handshake? Consider all cases.)

   Following the hint, let t represent a number of handshakes at the party so far, and Odd_t the number of people who, after the t'th handshake, have shaken hands with an odd number of people.

   We prove that Odd_t is even for all $t \geq 0$ by induction on t.

   First, for $t = 0$, no one has shaken hands. Thus, all people have shaken 0 hands, which is even, so $Odd_0 = 0$, which is even.

   Assume after t handshakes that Odd_t is even. Then the next handshake is either between two "evens", an "even" and an 'odd" or two "odds".

   In the first case, if two "evens" shake hands, they become "odds", so the number of odds has increased by 2, i.e. $Odd_{t+1} = Odd_t + 2$, which, since Odd_t is even, is still even.

   In the second case, if an "even" shakes hands with an "odd", then they switch categories, and the number of odds has not changed, i.e. $Odd_{t+1} = Odd_t$, which is even by the induction assumption.

   In the last case, if two "odds" shake hands, they become "evens", so the number of odds has decreased by 2, i.e. $Odd_{t+1} = Odd_t - 2$, which, since Odd_t is even, is still even.

   So in any case, the number of "odds" after the $t+1$'st shake, Odd_{t+1}, is even. Thus, by induction, Odd_t is even no matter how many shakes occured at the party. In particular, for t the total number of hand shakes at the party, Odd_t is even. Since I am "odd" , $Odd_t \geq 1$, and so must be ≥ 2 to be even. Therefore, someone else must also be "odd".

3. Consider the following pseudo-code program:

   (a) Program: LogLog
   (b) Input: x: Integer;
   (c) Output: i: Integer;
   (d) begin;
   (e) If $x \leq 1$ output "Undefined"
   (f) y:=2;
   (g) i:=0;
   (h) while $y \leq x$ do:

(i) begin;

(j) y:= y^2;

(k) i:=i+1;

(l) end;

(m) Return i;

(n) end;

Prove that on any input $x \geq 2$, the output of this program is $\lfloor \log_2(log_2(x) \rfloor$. (Hint: prove the following loop invariant: After t loops, $i_t = t$ and $y_t = 2^{2^t}$. Then show the smallest t so that the guard is false is $\lfloor \log_2 \log_2 x \rfloor$, using algebra.

: We'll follow the hint to show by induction on t, the number of loops, that $y_t = 2^{2^t}$ and $i_t = t$.

First, for $t = 0$,since y and i are initialized to 2 and 0, $y_0 = 2 = 2^1 = 2^{2^0}$ as claimed, and $i_0 = 0$ as claimed.

Assume that the claim is true after $t = k$ loops. Then $y_k = 2^{2^k}$ and $i_k = k$. After the next loop, we get $y_{k+1} = y_k^2 = (2^{2^k})^2 = 2^{2(2^k)} = 2^{2^{k+1}}$ as desired, and $i_{k+1} = i_k + 1 = k + 1$ as desired.

Thus, by induction, if the program loops t times, the values of i and y are given by $i_t = t, y_t = 2^{2^t}$. In particular, the guard fails after the first t so that $y_t > x$, i.e., if $2^{2^t} > x$. Taking logs twice,this is the same as $t > \log_2(\log_2 x)$. So the guard will fail after the first integer strictly greater than $\log_2 \log_2 x$.

At this point, I must apologize. The first integer *strictly greater* than an real number r isn't $\lfloor r \rfloor$ but $\lfloor r \rfloor + 1$. So my program is always off by 1. It actually returns $i_t = t = \lfloor \log_2 \log_2 x + 1$! I should subtract 1 from the final answer.

I only realized this by doing the proof, which does show why you need to verify code. (Of course, in this example, testing it out on a small number like 2 would have been helpful.)

I'll give 1 pt. extra credit (on the average for the course) to everyone who caught this bug.