CSE 20, Discrete Mathematics, Spring 1997
Sample Questions, Second Midterm
Chapter 4: Induction and its applications

Format: There will be three questions on the midterm. All involve proofs by induction. The first will be a numerical question (equality, inequality, divisibility), the second will be a program analysis (correctness argument or time analysis) and the last will be a "systems evolution" word problem. These are just all mixed together.

1. Prove that $\forall n, (n \geq 1) \rightarrow n^n \geq n!$.

   We prove this by induction on n. First, for $n = 1$, $1^1 = 1 \geq 1 = 1!$, so the inequality holds for $n = 1$.

   For the induction step, we show that if the inequality holds for $n = k$, then it also holds for $n = k + 1$. Let k be any positive integer, and assume $k^k \geq k!$ Then $(k+1)^(k+1) = (k+1)(k+1)^k \geq (k+1)k^k \geq (k+1)k! = (k+1)!$. So if the inequality holds for $n = k$, then it also holds for $n = k + 1$.

   Thus, since the inequality is true for $n = 1$, and if it is true for $n = k$, then it is also true for $n = k+1$, by the principle of induction it is true for all positive integers n.

2. A certain company, having accumulated a fair bit of debt, $D(0)$, has decided on the following austerity measure:

   - At the beginning of each year $t \geq 1$, the company first uses all the revenue $R(t-1)$ from the previous year to pay off as much debt as possible. Then the company will borrow enough money to pay back the rest of what is owing on the previous year's loan, and to cover its budget $B(t)$ for the year at interest rate $I(t)$. (Even if the company can pay back all its debt from the previous year, it will still need to borrow for its budget for the coming year.) Thus the amount of money it will owe at the end of the year is $D(t) = (1 + I(t))(D(t-1) - R(t-1) + B(t))$
   - All of last year's revenues, $R(0)$, are available so this policy can start in this year, $t = 1$.
   - For $t \geq 1$, the budget $B(t)$ will be set at no more than $.9R(t-1)$, 90% of the previous year's revenues.
   - The company assumes that the interest rate will never be more than 10%, and that their yearly revenues will never be less than their current debt, $D(0)$.

   Prove (using induction) that if the company follows the above policy, and its assumptions are always correct (e.g., correct about every year), then the company's debt at the end of any year will never be higher than it is now, at year 0, i.e $\forall t, D(t) \leq D(0)$.

   Solution: We are given the following:

   (a) $\forall t, (t \geq 1) \rightarrow [D(t) = (1 + I(t))(D(t-1) - R(t-1) + B(t))]$
   (b) $\forall t, (t \geq 1) \rightarrow [B(t) \leq .9R(t-1)]$
   (c) $\forall t (I(t) \leq .1)$
   (d) $\forall t (R(t) \geq D(0))$

   We want to show $\forall t (D(t) \leq D(0))$.

   We do this by induction on t For $t = 0$, the claim is true, $D(0) \leq D(0)$. Assume the claim is true for $t = k$, for k an arbitrary non-negative integer, i.e., $D(k) \leq D(0)$. Then $D(k+1) = (1 + I(k))(D(k) - R(k) + B(k+1)) \leq 1.1(D(k) - R(k) + B(k+1)) \leq 1.1(D(k) - R(k) + .9R(k)) = 1.1(D(k) - .1R(k)) \leq 1.1(D(k) - .1D(0)) \leq 1.1(D(0) - .1D(0)) = .99D(0) \leq D(0)$. (Reasons: premise 1, premise 3, premise 2, premise 4, induction hypothesis) Thus, if the claim is true for $n = k$, it is also true for $n = k + 1$.

   Therefore, since the claim is true for $n = 0$, and if it is true for $n = k$, it is also true for $n = k + 1$, by induction, it is true for all n.

3. Define the Fibonacci numbers by $F_0 = 0, F_1 = 1$, and $F_{i+2} = F_{i+1} + F_i$ for $i \geq 0$. Prove that for any $n \geq 1$, $\sum_{i=1}^{i=n} F_{2i} = F_{2n+1} - 1$.

   We prove this by (normal) induction on n For $n = 1$, $\sum_{i=1}^{i=1} F_{2i} = F_2 = 1 = 2 - 1 = F_3 - 1$, so the claim is true for $n = 1$. Assume the claim is true for $n = k$, i.e., $\sum_{i=1}^{i=k} F_{2i} = F_{2k+1} - 1$. Then $\sum_{i=1}^{i=k+1} F_{2i} = \sum_{i=1}^{i=k} F_{2i} + F_{2k+2} = F_{2k+1} - 1 + F_{2k+2} = F_{2k+3} - 1 = F_{2(k+1)+1} - 1$. So if the claim is true for $n = k$, it is also true for $n = k + 1$. By induction, the claim holds for all values of $n \geq 1$.

4. Prove that $3^n > n2^n$ for any integer $n \geq 1$ (Hint: Your induction step might only work for $n \geq 2$; if so, do both $n = 1$ and $n = 2$ as base cases.) For $n = 1$, we have $3^1 = 3 > 2 = 12^1$. For $n = 2$, $3^2 = 9 > 8 = 22^2$. Thus, the inequality is true for $n = 1, n = 2$ Assume the inequality holds for $n = k$ where $k \geq 2$. In other words, $3^k > k2^k$. Since $k \geq 2$, $3k \geq 2k + 2$. Then $3^{k+1} = 33^k > 3k2^k \geq (2k+2)2^k \geq (k+1)2^{k+1}$, so the inequality holds for $n = k + 1$. Since the inequality holds for $n = 1, 2$, and if it holds for any $n = k \geq 2$, then it also holds for $n = k + 1$, by the principle of induction, it holds for all positive integer values of n. Of course, I thought of this proof by looking at the goal $3^{k+1} > (k+1)2^{k+1}$ and simplifying, but the way I wrote the proof proceeds in a logical fashion from what we are given (the inequality holds for k) to what we want to prove (the inequality holds for $k+1$). So do calculations on scrap paper, and then write the proof in the order it makes sense to the reader.

5. There are $n \geq 3$ teams in a league, and each team played every other team. There were no ties, so for every two distinct teams, $x, y, x \neq y$ either x defeated y or y defeated x. Use induction to prove that either there is a team that defeated all other teams in the league, or there are three teams x, y, z so that x defeated y, y defeated z, and z defeated x. (Hint: one approach is to assume there are not three such teams, and then prove the following by induction on k: if we list the teams as $t_1, t_2, ..t_n$, then among the first k teams, there is always one team that defeated all the others in the first k.)

   Assume that there are no three distinct teams x, y, z so that x defeated y, y defeated z, and z defeated x. Let $t_1, t_2, ...t_n$ be a listing of the teams. We prove the following by induction on $k \geq 2$: $\exists i (1 \leq i \leq k) \wedge \forall j (1 \leq j \leq k \wedge j \neq i) t_i$ defeated t_j. Call such a team t_i the champion of the first k teams.

   For the base case, consider $k = 2$. Either t_1 defeated t_2 and t_1 is the champion of the first two teams, or t_2 defeated t_1, and t_2 is the champion of the first 2 teams.

   For the induction step, assume there is a champion t_i of the first k teams. We wish to show that there is a champion of the first $k + 1$ teams. Either t_i defeated t_{k+1} ,or t_{k+1} defeated t_i. In the first case, t_i is the champion of the first $k + 1$ teams. In the second case, we claim t_{k+1} is the champion of the first k teams. We prove the last claim by contradiction. Assume that t_{k+1} is not the champion of the first $k + 1$ teams. Then there is some team t_j with $1 \leq j \leq k$ so that t_{k+1} did not defeat t_j. $j \neq i$, since t_{k+1} defeated t_i. So t_i defeated t_j, since t_i was the champion of the first k teams. Then t_j defeated t_{k+1}, t_{k+1} defeated t_i, and t_i defeated t_j. This contradicts our assumption that there are no 3 teams forming a cycle. Therefore, our assumption that t_{k+1} was not the champion of the first $k + 1$ teams is incorrect, so in this case t_{k+1} is the champion of the first $k + 1$ teams. Thus, in either case, there is a champion for the first $k + 1$ teams.

   Thus, by the principle of induction, for all $2 \leq k \leq n$, there is a champion of the first k teams. In particular there is a champion of the first n teams, which is by definition a team that defeated all others in the league. So, if there are no three teams that defeat each other in a circle, then there is a champion, an undefeated team.

6. Consider the following pseudo-code program:

```
10 INPUT X,Y : INTEGER
15 C;=0
20 While X > Y , do:
30 begin
40 X:=(X+Y) DIV 2;
```

```
50 C:= C +1;
60 end
70 RETURN C
```

Prove that if the input values of X, Y are positive integers n and m, with $n > m$, then the program loops no more than $log_2(n-m)+1$ times. (Hint: for X_t the value of X after t loops, prove the following loop invariant: $X_t - m \leq (n-m)/2^t$. Then since X is an integer, if $t > log_2(n-m)$, then $X_t - m < 1$ so $X_t \leq m$)

Actually, I'll do the proof by strong induction on X instead. Let y be a fixed integer. Let $T(x, y)$ be the number of loops the program makes when $X = x, Y = y$. We wish to prove that for all $x \geq y+1$, $T(x, y) \leq log_2(x-y)+1$..

Since we want to prove this for $x \geq y+1$, the base case is $x = y+1$. In this case, the program will loop once after which the value of Y is unchanged, and the value of X is $\lfloor (y+1+y)/2 \rfloor = \lfloor (y+1/2) \rfloor = y$. Thus, the guard will fail and the program halts. So $T(y+1, y) = 1 \leq log_2(1)+1 = log_2(y+1-y)+1$.

For the induction step, let $k > y+1$ and assume $T(x, y) \leq log_2(x-y)+1$ for all $y+1 \leq x \leq k-1$. Then on input k, y, after the program loops once, the new value of X is $\lfloor (k+y)/2 \rfloor$. Since $y+2 \leq k$, we have $\lfloor (k+y)/2 \rfloor \geq \lfloor (2y+2)/2 \rfloor = y+1$ and we have $\lfloor (k+y)/2 \rfloor \leq (k+k-2)/2 \rfloor = k-1$. So we can apply the induction hypothesis to this value of X to get that $T(k, y) = 1 + T(\lfloor (k+y)/2 \rfloor, y) \leq 1 + log_2(\lfloor (k+y)/2 \rfloor - y) + 1 \leq log_2((k+y)/2 - y) + 2 = log_2((k-y)/2) + 2 = log_2(k-y) - 1 + 2 = \log_2(k-y) + 1$, which is what we wanted.

Thus, by strong induction, for all $x \geq y+1$, $T(x, y) \leq \log_2(x-y)+1$. Since y was an arbitrary integer, this is true for all y's. and all $x > y$.

7. Consider the following pseudo-code program:

```
10 INPUT X : INTEGER
15 C:=2
20 While X > 1 , do:
30 begin
40 X:=X DIV C;
50 C:= C +1;
60 end
70 RETURN C
```

Prove that if the input value of X is a positive integer n, then the program loops no more than $log_2(n)$ times. (Hint: for X_t the value of X after t loops, prove the following loop invariant: $X_t \leq n/2^t, C_t \geq 2$. Then since X is an integer, if $t \geq log_2(n)$, then $X_t \leq 1$)

This time we'll follow the hint. The idea is we're at least halving X each time we go through the loop.

Let X_t and C_t be the values of X, C after t loops. We claim that $X_t \leq n/2^t$ and $C_t \geq 2$.

First, the base case is $t = 0$, i.e. for the values before the loop begins. After 0 times through the loop, the variables are initialized to $X_0 = n \leq n/1 = n/2^0$, and $C_0 = 2 \geq 2$. So both parts of the claim are true before the loop starts.

Second, assume both parts of the claim are true after $t = k$ loops for some integer $k \geq 0$, i.e., that $X_k \leq n/2^k$ and $C_k \geq 2$. Then after one more loop, $X_{k+1} = \lfloor X_k/C_k \rfloor \leq X_k/C_k \leq X_k/2 \leq (1/2)(n/2^k) = n/2^{k+1}$, so the first part of the claim is true for $t = k+1$. Also $C_{k+1} = C_k + 1 \geq 2+1 \geq 2$, so the second part is also true.

Thus, by induction both parts of the claim are true after any number of loops t.

In particular, if the program loops t times and $t \geq log_2 n$, then the value of X after the t loops is $X_t \leq n/2^t \leq n/2^{\log_2 n} = n/n = 1$. So after t loops, the guard will fail, and the program will halt. Thus, the program never loops more than $\log_2 n$ times.

(Note that we didn't need the second part of the claim in the final argument. However, the induction step wouldn't be valid without it. Sometimes to prove one thing by induction, you actually need to prove something more, so that the induction step works.)

8. Consider the following "pseudo-code" program.

   Program:DV
   Input: X, Y (Integer)
   begin;
   Q : = 0;
   While (X$\geq$ Y) do:
   begin{while};
   X: = X-Y
   Q:= Q+1
   end{while};
   Output Q;
   end;

   Prove that on positive integeral inputs $X = n, Y = m$, that the output of this program is $\lfloor n/m \rfloor$. Hint: Loop invariant method: What will the values of X and Q be after t loops? Prove your answer correct by induction on t.

   Following the hint, we'll see that after t loops, the value of X is $X_t = n - tm$ and the value of Q is $Q_t = t$.

   We'll prove this by induction.

   First, for $t = 0$, the variables are initialized to $X_0 = n = n - 0m$ and $Q_0 = 0$. So both parts of the claim is true for $t = 0$,

   Assume for some $k \geq 0$ that $X_k = n - km$ and $Q_k = k$. If the program loops once more, then $X_{k+1} = X_k - m = n - km - m = n - (k+1)m$ and $Q_{k+1} = Q_k + 1 = k + 1$. So both parts of the claim are true after $t = k + 1$ loops.

   Thus, by induction, for any $t \geq 0$, $X_t = n - tm$ and $Q_t = t$.

   Thus, the guard will fail after t steps if and only if $X_t = n - tm < m$ or dividing by m and adding t, if $n/m < t + 1$ In other words, the program halts at the smallest integer t with $t > n/m - 1$. This is $t = \lfloor n/m \rfloor$, since $\lfloor n/m \rfloor > n/m - 1 \geq \lfloor n/m \rfloor - 1$. Thus, when $t = \lfloor n/m \rfloor$ the program halts, and it outputs $Q_t = t = \lfloor n/m \rfloor$.

9. An elevator started and ended the day on the ground floor. Prove that the number of times it went from the third floor to the fourth floor equals the number of times it went down from the fourth floor to the third floor. For the purposes of this problem, we think of the elevator as going from the third to the fourth whenever it starts at the third floor or any floor below the third, and ends at the fourth or higher floor, whether or not it stops at the third and fourth floors. (Hint: Consider the day in the life of the elevator as a sequence of n trips, from one floor to another floor. Let u_t be the number of times the elevator has gone up from the third to fourth floors after t trips, d_t the number of times it has gone down from the fourth to the third; and f_t be the floor that the t'th trip ends in. Prove by induction that $u_t - d_t = 0$ if $f_t \leq 3$, and $u_t - d_t = 1$ if $f_t \geq 4$. Then apply this for $t = n$; we are given $f_n = 1$ (the last trip ends on the first floor.)

   Following the hint, we'll prove that if $f_t \geq 4$ then $u_t = d_t + 1$, and otherwise $u_t = d_t$.

   First, for $t = 0$, at the start of the day, $f_0 = 1 < 4$ and $u_0 = d_0 = 0$. So the claim is true after $t = 0$ trips.

   Assume the claim is true after $t = k$ trips. There are four cases: Case 1: The $k + 1$'st trip began and ended at a floor at most 3. Then, $f_k \leq 3, f_{k+1} \leq 3$, and the values of $u_{k+1} = u_k$ and $d_{k+1} = d_k$ don't change. Thus, since the claim is true for $t = k$ and $f_k \leq 3$, $u_k = d_k$ and hence $u_{k+1} = u_k = d_k = d_{k+1}$. Since $f_{k+1} \leq 3$, this is what we wanted.

Case 2: The $k+1$'st trip began at a floor at most 3 and ended at a floor at least 4. Then, $f_k \leq 3, f_{k+1} \geq 4$, and the value of $u_{k+1} = u_k + 1$ increases by 1 and $d_{k+1} = d_k$ doesn't change. Thus, since the claim is true for $t = k$ and $f_k \leq 3$, $u_k = d_k$ and hence $u_{k+1} = u_k + 1 = d_k + 1 = d_{k+1} + 1$. Since $f_{k+1} \geq 4$, this is what we wanted.

Case 3: The $k+1$'st trip began at least at floor 4 and ended at a floor at most 3. Then, $f_k \geq 4, f_{k+1} \leq 3$, and the value of $u_{k+1} = u_k$ doesn't change, but $d_{k+1} = d_k + 1$ increases. Thus, since the claim is true for $t = k$ and $f_k \geq 4$, $u_k = d_k + 1$ and hence $u_{k+1} = u_k = d_k + 1 = d_{k+1}$. Since $f_{k+1} \leq 3$, this is what we wanted.

Case 4: The $k+1$'st trip began and ended at a floor at least 4. Then, $f_k \geq 4, f_{k+1} \geq 4$, and the values of $u_{k+1} = u_k$ and $d_{k+1} = d_k$ don't change. Thus, since the claim is true for $t = k$ and $f_k \geq 4$, $u_k = d_k + 1$ and hence $u_{k+1} = u_k = d_k + 1 = d_{k+1} + 1$. Since $f_{k+1} \geq 4$, this is what we wanted.

Thus, in all cases, the claim is true for $t = k+1$, and hence by induction for all t. In particular, for t the total number of trips the elevator made that day, we know that $f_t = 1 < 4$ since the elevator ended at the ground floor. Then the claim tells us $u_t = d_t$, i.e. the total number of passes going up equals that coming down.

10. Consider the following pseudo-code program:

    (a) Program: Log; Input:X=Integer; Output:L=Integer;
    (b) begin;
    (c) L:=0;
    (d) C:=1;
    (e) While $X \geq 2 * C$ do:
    (f) begin;
    (g) L:=L+1;
    (h) C:=2*C;
    (i) end;
    (j) Return L;
    (k) end;

    Prove that for any $n \geq 1$, on input $X = n$, *Log* returns value $\lfloor log_2 n \rfloor$, (the greatest integer part of the log base 2 of n.) (Hint: Use the loop-invariant method to analyse C_t, the value of C after t loops and L_t, the value of L after t loops. Use this to calculate the smallest value of t that causes the program to get out of the loop.)

    This is very similar to the last program.