Definitions
Below are definitions that will be used in the problem statements. fact, app, etc. are omitted here since they are given in the homework.

;;; choose : nat x nat -> nat
;;; Binomial choose function
(defun choose (n k)
  (/ (fact n) (* (fact k) (fact (- n k)))))

;;; posp : any -> Boolean
;;; T iff given a positive integer
(defun posp (n)
  (not (zp n)))
Problems

Prove the following theorems. If you use induction, clearly indicate what functions were used to generate the induction schemes.

1. (implies (and (posp n) (posp k) (< k n))
   (equal (+ (choose (- n 1) (- k 1)) (choose (- n 1) k)) (choose n k)))

   The above is not a theorem without the hypotheses. Why not?

2. (implies (and (natp n) (natp k) (<= k n))
   (natp (choose n k)))

3. (booleanp (in a X))

4. (implies (and (not (= a b)) (in a (rem-el b X)))
   (in a X))

5. (implies (and (not (= a b)) (in a X))
   (in a (rem-el b X)))

6. (implies (not (= a b)) (= (in a (rem-el b X)) (in a X)))

7. (implies (in a X) (not (in a (diff Y X))))

8. (implies (consp X) (not (<= X (diff Y X))))
Solutions

1. We solve this with equational reasoning.

Context:
-----------
(1) (posp n)
(2) (posp k)
(3) (< k n)
-----------

(+ (choose (~ n 1) (~ k 1)) (choose (~ n 1) k))
= {def. choose}
(+ (/* (fact (~ n 1))
    (* (fact (~ n 1)) (fact (~ n 1) (~ k 1)))
    (/* (fact (~ n 1))
      (* (fact k) (fact (~ n 1) (~ k 1)))))
= {def. fact, (2) commutativity of *, (a-1)-b=(a-b)-1}
(+ (/* (k (fact (~ n 1)))
    (* (fact k) (fact (~ n k)))
    (/* (~ n k) (fact (~ n 1)))
    (* (fact k) (~ n k) (fact (~ n k 1)))))
= {def. fact, (3)}
(+ (/* (k (fact (~ n 1)))
    (* (fact k) (fact (~ n k)))
    (/* (~ n k) (fact (~ n 1)))
    (* (fact k) (fact (~ n k))))
= {arith. a/b = (a*c)/(b*c), c!=0, (2), (3)}
(+ (/* (k (fact (~ n 1)))
    (* (fact k) (fact (~ n k)))
    (/* (~ n k) (fact (~ n 1)))
    (* (fact k) (~ n k) (fact (~ n k 1)))))
= {def. fact, (3)}
(+ (/* (k (fact (~ n 1)))
    (* (fact k) (fact (~ n k)))
    (/* (~ n k) (fact (~ n 1)))
    (* (fact k) (fact (~ n k))))
= {arith. a/c + b/c = (a+b)/c}
(/* (/* (k (fact (~ n 1)))
    (* (~ n k) (fact (~ n 1)))
    (* (fact k) (fact (~ n k))))
= {arith. distributivity, canceling}
(/* (~ n (fact (~ n 1)))
    (* (fact k) (fact (~ n k))))
= {def. fact, (1)}
(/* (fact n)
    (* (fact k) (fact (~ n k))))
= {def. choose}
(choose n k)
Q.E.D.
2. We prove this with induction on $n$, given the result of 3. in the lab questions (call this fact-nat lemma). In this case, we have the induction hypothesis stated twice, giving two substitutions for the free variable $k$.

Induction scheme:

\[
\begin{align*}
& (\text{natp } k) \\
& (zp \ n) \\
& (\leq k \ n) \\
& (\text{natp} (\text{choose } n \ k)) \\
\end{align*}
\]

\[
\begin{align*}
& (\text{natp } k) \\
& (\leq k \ n) \\
& (\text{implies} (\text{natp} (- n \ 1)) \\
& (\text{natp } k) \\
& (\leq k (- n \ 1)) \\
& (\text{natp} (\text{choose} (- n \ 1) \ k)) \\
\end{align*}
\]

\[
\begin{align*}
& (\text{implies} (\text{natp} (- n \ 1)) \\
& (\text{natp} (- k \ 1)) \\
& (\leq (- k \ 1) (- n \ 1)) \\
& (\text{natp} (\text{choose} (- n \ 1) (- k \ 1)))) \\
\end{align*}
\]

\[
\begin{align*}
& (\text{natp} (\text{choose} n \ k)) \\
\end{align*}
\]

Base Case:

Context:

\[
\begin{align*}
& (1) \ (\text{natp } k) \\
& (2) \ (zp \ n) \\
& (3) \ (\leq k \ n) \\
& (4) \ (zp \ k) \ \{(1) \ (2) \ (3)\} \\
\end{align*}
\]

\[
\begin{align*}
& \{\text{def. choose, def. fact, (2), (4)}\} \\
& \text{(natp 1)} \\
& \{1 \text{ is a natural}\} \\
& T
\end{align*}
\]
Induction Step:
Context:
---------------
(1) \((\neg (zp n))\)
(2) \((natp k)\)
(3) \((\leq k n)\)
(4) \((\text{implies} \ (\text{and} \ (natp (- n 1)))\)
    \(\ (natp k)\)
    \(\ (\leq k (- n 1))\)
    \(\ (natp (\text{choose} (- n 1) k)))\)
(5) \((\text{implies} \ (\text{and} \ (natp (- n 1)))\)
    \(\ (natp (- k 1))\)
    \(\ (\leq (- k 1) (- n 1))\)
    \(\ (natp (\text{choose} (- n 1) (- k 1))))\)
(6) \((natp n) \{1\}\)
(7) \((< 0 n) \{1\}\)
(8) \((natp (- n 1)) \{6, 7\}\)
(9) \((\leq (- k 1) (- n 1)) \{3, \text{arithmetic}\}\)
(A) \((\text{implies} \ (\leq k (- n 1))\)
    \(\ (natp (\text{choose} (- n 1) k)))\) \(\{2, 4, 8\}\)
(B) \((\text{implies} \ (natp (- k 1))\)
    \(\ (natp (\text{choose} (- n 1) (- k 1))))\) \(\{5, 8, 9\}\)
(C) \((= (natp (- k 1)) (\text{posp} k)) \{2\}\)
(D) \((= (\leq k (- n 1)) (< k n)) \{2\}\)
---------------

[Case Split]
  C1. \((= k n)\)
      \(\ (natp (\text{choose} n k))\)
      \(= \{\text{def. choose, (C1)\}\}
      \(\ (natp (/ (\text{fact} n) (* (\text{fact} n) (\text{fact} (- n n))))))\)
      \(= \{\text{def. fact, fact-nat lemma\}\}
      \(\ (natp 1)\)
      \(= \{1 \text{ is a natural\}\}
      T

  C2. \((\neg (= k n))\)
[Case Split]
  C3. \((= k 0)\)
      \(\ (natp (\text{choose} n k))\)
      \(= \{\text{def. choose, (C3)\}\}
      \(\ (natp (/ (\text{fact} n) (* (\text{fact} 0) (\text{fact} (- n 0))))))\)
      \(= \{\text{def. fact, fact-nat lemma\}\}
      \(\ (natp 1)\)
      \(= \{1 \text{ is a natural\}\}
      T
C4. (not (= k 0))
   (natp (choose n k))
= {Theorem 1, (C4), def. posp, (1), (2), (3)}
   (natp (+ (choose (- n 1) (- k 1)) (choose (- n 1) k)))
= {Arithmetic}
   (and (natp (choose (- n 1) (- k 1)))
       (natp (choose (- n 1) k)))
= {(C4), (2), (B)}
   (and T
       (natp (choose (- n 1) k)))
= {(3), (C2), (A)}
   (and T
       T)
= {Propositional Logic}
   T
Q.E.D.
Induct on (true-listp X) with scheme
(and (implies (endp X)
          (booleanp (in a X)))
   (implies (and (consp X)
                  (booleanp (in a (cdr X))))
          (booleanp (in a X))))

Base case:
Context:
-------------------
(1) (endp X)
-------------------
   (booleanp (in a X))
   = {def. in, (1)}
   (booleanp nil)
   = {def. booleanp}
   T

Induction Step:
Context:
------------------
(1) (consp X)
(2) (booleanp (in a (cdr X)))
------------------
   (booleanp (in a X))
   = {def. in, (1)}
   (booleanp (or (= a (car X))
                (in a (cdr X))))
   = {= is boolean, (2), or of booleans is boolean}
   T
Q.E.D.
We induct on (true-listp X) with scheme
(and (implies (and (endp X)
  (not (= a b))
  (in a (rem-el b X)))
  (in a X))
(implies (and (consp X)
  (not (= a b))
  (in a (rem-el b X))
  (implies (and (not (= a b))
    (in a (rem-el b (cdr X)))
    (in a (cdr X))))
  (in a X)))

4. Base Case:
Context:
---------------------
(1) (endp X)
(2) (not (= a b))
(3) (in a (rem-el b X))
(4) (in a nil) {1, 3, def. rem-el}
(5) nil {4, def. in}
---------------------
(in a X)
= {Propositional Logic}
T
Induction Step:
Context:

-------------------
(1) (consp X)
(2) (in a (rem-el b X))
(3) (not (= a b))
(4) (implies (and (not (= a b))
    (in a (rem-el b (cdr X))))
    (in a (cdr X)))
(5) (implies (in a (rem-el b (cdr X)))
    (in a (cdr X))) {3, 4}

-------------------
(in a X)
= {def. in, (1)}
(or (= a (car X))
    (in a (cdr X)))

[Case Split]

Context:

-------------------
(C1) (= b (car X))
(6) (in a (rem-el (cdr X)) {1, 2, C1, def. rem-el}
(7) (in a (cdr X)) {5, 6}

-------------------
(or (= a (car X))
    (in a (cdr X)))

= {(3), (7), Propositional Logic}

T

Context:

-------------------
(C2) (not (= b (car X)))
(6) (in a (cons (car X) (rem-el b (cdr X)))) {1, 2, C2, def. rem-el}
(7) (or (= a (car X))
    (in a (rem-el b (cdr X)))) {6, def. in}

-------------------

9
[Case Split]

Context:

(C3) (= a (car X))

(or (= a (car X))
  (in a (cdr X)))

= {(C3), Propositional Logic}

T

Context:

(C4) (not (= a (car X)))
(8) (in a (rem-el b (cdr X))) {C4, 7}
(9) (in a (cdr X)) {5, 8}

(or (= a (car X))
  (in a (cdr X)))

= {(9), Propositional Logic}

T

Q.E.D.
We induct on (true-listp X) with scheme
(and (implies (and (endp X)
    (not (= a b))
    (in a X))
    (in a (rem-el b X)))
(implies (and (consp X)
    (not (= a b))
    (in a X)
    (implies (and (not (= a b))
      (in a (cdr X)))
      (in a (rem-el b (cdr X))))))

5. Base Case:
Context:
------------
(1) (endp X)
(2) (not (= a b))
(3) (in a X)
(4) nil {def. in, 1}
------------
(in a (rem-el b X))
= {Propositional Logic}
  T
Induction step:
Context:

-----------------------
(1) (consp X)
(2) (not (= a b))
(3) (in a X)
(4) (implies (and (not (= a b))
   (in a (cdr X)))
   (in a (rem-el b (cdr X))))
(5) (implies (in a (cdr X))
   (in a (rem-el b (cdr X)))) {2}
----------------------

[Case Split]
Context:

----------------------
(C1) (= a (car X))
----------------------

= {def. rem-el, (1), (2)}
   (in a (cons (car X) (rem-el b (cdr X))))
= {def. in, car/cons axiom, (C1)}
   (or (= a a)
      (in a (rem-el b (cdr X))))
= {reflexivity axiom}
   T

Context:

----------------------
(C2) (not (= a (car X))
(6) (in a (cdr X)) {C2, 3}
(7) (in a (rem-el b (cdr X))) {5, 6}
----------------------

= {def. rem-el, (1)}
   (in a (if (= b (car X))
      (rem-el b (cdr X))
      (cons (car X) (rem-el b (cdr X)))))
= {if lifting}
   (if (= b (car X))
      (in a (rem-el b (cdr X)))
      (in a (cons (car X) (rem-el b (cdr X)))))
= {def. in, car/cons axiom, (2)}
   (if (= b (car X))
      (in a (rem-el b (cdr X)))
      (or nil (in a (rem-el b (cdr X)))))
= {propositional logic}
   (in a (rem-el b (cdr X)))
= {(7)}
   T
Q.E.D.
(implies (not (= a b))
   (= (in a (rem-el b X))
       (in a X)))

Context:
------------------
(1) (not (= a b))
------------------

(= (in a (rem-el b X))
    (in a X))
= {in is boolean, produces two subgoals:}
SG1: (implies (in a (rem-el b X))
        (in a X))
SG2: (implies (in a X)
        (in a (rem-el b X)))
Each are discharged by the previously proved theorems with (1)
Q.E.D.
(implies (in a X)  
  (not (in a (diff Y X))))

Induct on (true-listp X) with scheme
(implies (and (endp X)  
  (in a X))  
  (not (in a (diff Y X))))
(implies (and (consp X)  
  (in a X)  
  (not (in a (diff Y (cdr X)))))  
  (not (in a (diff Y X))))

Base Case:
Context:
--------------------
(1) (endp X)  
(2) (in a X)  
(3) nil {1, 2, def. in}
------------------------
(not (in a (diff Y X)))
= {Propositional Logic}  
  T

7. Induction Step
Context:
--------------------
(1) (consp X)  
(2) (in a X)  
(3) (not (in a (diff Y (cdr X)))))
---------------
(not (in a (diff Y X)))
= {def. diff, (1)}  
  (not (in a  
    (rem-el (car X) (diff Y (cdr X)))))

[Case Split]
  C1. (= a (car X))
  = {(C1), Q5}  
    T
  
  C2. (not (= a (car X)))
  = {Above theorem, C2}  
    (not (in a (diff Y (cdr X)))))
  = {(3)}  
    T
Q.E.D.
Context:
---------------
(1) (consp X)
---------------
(not (=< X (diff Y X)))
= {def. =<, (1)}
(not (and (in (car X) (diff Y X))
 (=< (cdr X) (diff Y X))))
S. = {Propositional logic}
(or (not (in (car X) (diff Y X)))
 (not (=< (cdr X) (diff Y X))))
= {Above theorem, (in (car X) X) = t}
(or t
 (not (=< (cdr X) (diff Y X))))
= {Propositional logic}
T
Q.E.D.