

Homework 3:

- You *have to* work in pairs for this homework and you have to work with someone you have not worked with yet. Send the name of your partner to Zhifeng (austin@ccs.neu.edu) by the end of the day, on Wednesday Feb. 13th. If you do not do this, you will get a 0 on your homework.
- Each pair should submit their homework via blackboard. Each pair should submit just once. That is, one person per team should submit the homework via their blackboard account. Your submission should be a text file. The format of the file should be:
 - the names of the team
 - clear solutions to the problems

PROBLEM 1

Which of the following are Boolean tautologies? If they are not tautologies, exhibit a counterexample. If they are tautologies, give an informal justification.

- (a) $[(x \wedge y) \Rightarrow z] \Leftrightarrow [(\sim x \vee \sim y) \Rightarrow \sim z]$
- (b) $[(x \Leftrightarrow \sim y) \Leftrightarrow z] \Leftrightarrow [x \Leftrightarrow \sim(y \Leftrightarrow z)]$
- (c) $[(x \Rightarrow \sim x) \Rightarrow (\sim x \Rightarrow x)]$
- (d) $[(x \wedge y \wedge w \wedge v) \Rightarrow z] \Leftrightarrow [z \vee \sim v \vee \sim x \vee \sim y \vee \sim w]$

END PROBLEM 1

PROBLEM 2

Are the following theorems? If so, give a proof. If not, exhibit a counterexample.

- (a)
 - (atom x)
 - \Rightarrow (equal (append x nil) x)
- (b)
 - (atom x)
 - \Rightarrow (equal (append nil x) x)

END PROBLEM 2

PROBLEM 3

- (a) Define a function, filter-negative-integers that given a true-list of integers filters out all the negative integers, i.e., it returns the list obtained by removing the the negative integers from the input list.
- (b) Formalize the conjecture that (filter-negative-integers l) returns a list of positive integers, if the input (l) is a true-list of integers.
- (c) Is the conjecture true or false? If false, exhibit a counterexample. If true, briefly describe why.

END PROBLEM 3

PROBLEM 4

Consider the following function:

```
(defun rev (l)
  (if (endp l)
      nil
      (append (rev (cdr l))
               (list (car l))))))
```

Prove the following theorems:

(a)

```
(implies (endp l)
          (true-listp (rev l)))
```

(b)

```
(implies (and (consp l)
              (true-listp (cdr l)))
          (true-listp (rev l)))
```

You can use the following lemma:

```
(implies (true-listp y)
          (true-listp (append x y)))
```

PROBLEM 5

Use the definition of rev, above. Also, assume that the following are axioms:

```
1: (true-listp (rev x))
2: (implies (true-listp x)
            (equal (append x nil) x))
3: (equal (append (append x y) z)
          (append x (append y z)))
```

Prove the following:

(a)

```
(implies (endp x)
          (equal (rev (append x y))
                 (append (rev y) (rev x))))
```

(b)

```
(implies (and (consp x)
              (equal (rev (append (cdr x) y))
                     (append (rev y) (rev (cdr x)))))
          (equal (rev (append x y))
                 (append (rev y) (rev x))))
```

END PROBLEM 5

PROBLEM 6

Consider the following function:

```
(defun sum1 (n)
  (if (zp n)
```

```

0
(+ (* 3 n n) (* -3 n) 1 (sum1 (- n 1))))

```

Prove the following theorems:

(a)

```

(implies (and (zp n) (natp n))
  (equal (sum1 n) (* n n n)))

```

(b)

```

(implies (and (natp n)
  (> n 0)
  (equal (sum1 (- n 1))
    (* (- n 1) (- n 1) (- n 1))))
  (equal (sum1 n)
    (* n n n)))

```

END PROBLEM 6

PROBLEM 7

Consider the following function:

```

(defun sum2 (n)
  (if (zp n)
    0
    (+ (expt (* 2 n) 2)
      (sum2 (- n 1)))))

```

Prove the following theorems:

(a)

```

(implies (and (natp n) (zp n))
  (equal (sum2 n)
    (/ (* 2 n (+ n 1) (+ 1 (* 2 n))) 3)))

```

(b)

```

(implies (and (natp n) (> n 0)
  (equal (sum2 (- n 1))
    (/ (* 2 (- n 1) n (+ 1 (* 2 (- n 1))) 3))))
  (equal (sum2 n)
    (/ (* 2 n (+ 1 n) (+ 1 (* 2 n))) 3)))

```

END PROBLEM 7

PROBLEM 8

Consider the following function:

```

(defun sum3 (n)
  (if (zp n)
    0
    (+ (* n n) (sum3 (1- n)))))

```

Prove the following theorems:

(a)

```

(implies (and (natp n) (zp n))
  (equal (sum3 n)
    (/ (* n (+ n 1) (+ 1 (* 2 n)))
      6)))

```

(b)

```
(implies (and (natp n) (> n 0)
  (equal (sum3 (- n 1))
    (/ (* (- n 1) n (+ 1 (* 2 (- n 1))))
      6)))
  (equal (sum3 n)
    (/ (* n (+ n 1) (+ 1 (* 2 n)))
      6)))
```

END PROBLEM 8

**PROBLEM 9

Consider the following definition of pascal's triangle:

```
(defun pascal (x y)
  (if (or (zp x)
    (zp y)
    (<= x y))
    1
    (+ (pascal (- x 1) y)
      (pascal (- x 1) (- y 1)))))
```

Recall also the definition of factorial:

```
(defun fact (x)
  (if (zp x)
    1
    (* x (fact (- x 1)))))
```

Now, either falsify or prove the following conjectures.

(a)

```
(implies (and (natp x) (natp y) (>= x y)
  (or (zp x) (zp y) (<= x y)))
  (equal (pascal x y)
    (/ (fact x) (* (fact y) (fact (- x y))))))
```

(b)

```
(implies (and (natp x) (natp y) (>= x y)
  (not (zp x)) (not (zp y)) (< y x)
  (equal (pascal (- x 1) (- y 1))
    (/ (fact (- x 1))
      (* (fact (- y 1)) (fact (- (- x 1) (- y 1))))))
  (equal (pascal (- x 1) y)
    (/ (fact (- x 1))
      (* (fact y) (fact (- (- x 1) y))))))
  (equal (pascal x y)
    (/ (fact x)
      (* (fact y) (fact (- x y))))))
```

**END PROBLEM 9