A Sample Proof Using English and Program Reductions
COMP 280, Spring 2000

This example demonstrates inductive proofs that involve a combination of English text and program manipulation. Each case (base, inductive, and subcases thereof) in such a proof contains two steps:

1. Clearly state (in English) the expected result of running the program.
2. Use program reductions to derive an expression whose value matches the expected result.

The following sample proof that the *length* program correctly counts the number of items in a list illustrates these steps. This same technique applies to all inductive proofs that involve a combination of reasoning in English and reasoning with equations or reductions.

A list is either
- *empty*
- (*cons E L*) where *E* is an element and *L* is a list

```
;; length : list → number
;; returns the count of items in the list
(define (length a-lst)
  (cond [(empty? a-lst) 0]
        [else (+ 1 (length (rest a-lst)))]))
```

Prove that for all lists *L*, (length L) returns the number of items in the list.

Proof:
Base: Let *L* be *empty*. As the empty list contains no elements, we must prove that (*length empty*) returns 0 [*note: this is the expected result*].

```
    (length empty)
=
    (cond [(empty? empty) 0]
          [else (+ 1 (length (rest empty)))])
=
    0
```

Assumption: Let *L1* be a list. Assume that *length* correctly counts the number of items in *L1*.

Inductive: We must prove that *length* correctly counts the number of items in (*cons E L1*). Since (*cons E L1*) contains one more item than *L1*, we expect length to return one more than the number of items in *L1* [*note: this is the expected result*].

```
    (length (cons E L1))
=
    (cond [(empty? (cons E L1)) 0]
          [else (+ 1 (length (rest (cons E L1))))])
=
    (+ 1 (length (rest (cons E L1))))
=
    (+ 1 (length L1))
```

By assumption, (*length L1*) returns the number of items in *L1*. (+ 1 (*length L1*)) therefore returns one more than the number of items in *L1*, which matches our expected result.