Most of today’s lecture followed the *Sorting Integer Lists* notes from Mike Erdmann. We did the correctness proofs a little differently, so the versions we did are reproduced here.

**Lemma 1.** For all lists \( a \) and \( b \), if \( a \) sorted and \( b \) sorted, then \( \text{merge} (a, b) \) sorted and \( \text{merge} (a, b) \) is a permutation of \( a@b \).

**Proof.** By simultaneous induction on \( a \) and \( b \).

**Base case:** \((\[\], b)\). To show: \( \text{merge} (\[\], b) \) sorted and \( \text{merge} (\[\], b) \) is a permutation of \( \[\]@b \).

Proof: \( \text{merge} (\[\], b) \cong b \) by clause 1 of \( \text{merge} \). By assumption, \( a \) sorted and by definition, \( b \) is a permutation of \( \[\]@b \ (\cong b \ by \ @) \).

**Base case:** \((a, \[\])\). To show: \( \text{merge} (a, \[\]) \) sorted and \( \text{merge} (a, \[\]) \) is a permutation of \( a@\[\] \).

Proof: \( \text{merge} (a, \[\]) \cong a \) by clause 1 of \( \text{merge} \). By assumption, \( a \) sorted and by definition. By correctness of \( @ \), \( a@\[\] \cong a \), and \( a \) is a permutation of \( a \).

**Inductive step:** \((x :: a, y :: b)\). To show: \( \text{merge} (x :: a, y :: b) \) sorted and \( \text{merge} (x :: a, y :: b) \) is a permutation of \( (x :: a)@(y :: b) \).

IH1: \( \text{merge} (a, y :: b) \) sorted and \( \text{merge} (a, y :: b) \) is a permutation of \( a@(y :: b) \). IH2: \( \text{merge} (x :: a, b) \) sorted and \( \text{merge} (x :: a, b) \) is a permutation of \( (x :: a)@b \). IH3: \( \text{merge} (a, b) \) sorted and \( \text{merge} (a, b) \) is a permutation of \( a@b \).

**Proof:** We split into cases on \( \text{compare} (x, y) \).

**Case LESS.** \( \text{merge} (x :: a, y :: b) \cong x :: \text{merge} (a, y :: b) \) by the first branch of the case.

By IH1, \( \text{merge} (a, y :: b) \) sorted. Since \( \text{compare} (x, y) \cong \text{true} \) and \( x :: a \) sorted, \( x :: \text{merge}(a, y :: b) \) sorted. By IH1, \( \text{merge} (a, y :: b) \) is a permutation of \( a@(y :: b) \).

Thus, \( x :: \text{merge} (a, y :: b) \) is a permutation of \( (x :: a)@(y :: b) \).

The cases for \( \text{EQUAL} \) and \( \text{GREATER} \) are similar.

**Theorem 1.** For all lists \( l \), \( \text{msort} l \) sorted and \( \text{msort} l \) is a permutation of \( l \).

**Proof.** By strong induction on the structure of \( l \).

**Base case:** \([\]\). To show: \( \text{msort} [] \) sorted and \( \text{msort} [] \) is a permutation of \( [] \).

Proof: \( \text{msort} [] \cong [] \) by clause 1 of \( \text{msort} \).
**Base case:** $[x]$. To show: $\text{msort } x :: []$ sorted and $\text{msort } x :: []$ is a permutation of $x :: []$.

Proof: $\text{msort } x :: [] \cong x :: []$ by clause 2 of $\text{msort}$.

**Inductive step:** $l$. IH: For all sublists $l'$ of $l$, $\text{msort } l' \text{ sorted}$ and $\text{msort } l'$ is a permutation of $l'$.

To show: $\text{msort } l \text{ sorted}$ and $\text{msort } l$ is a permutation of $l$.

Proof: $\text{msort } l \cong \text{let val (A, B) = split } l \text{ in } \text{merge}(\text{msort } A, \text{msort } B) \text{ end}$ by the third clause of $\text{msort}$. By the lemma shown in the other set of notes, $\text{split } l \implies (a, b)$ and $a@b$ is a permutation of $l$ (so $a$ and $b$ are sublists of $l$).

By another step, $\text{msort } l \cong \text{merge } (\text{msort } a, \text{msort } b)$. By two applications of IH, this is equivalent to $\text{merge } (a', b')$ where $a' \text{ sorted}$, $b' \text{ sorted}$ and $a'$ and $b'$ are permutations of $a$ and $b$ respectively. By Lemma 1, this is equivalent to $l'$, where $l' \text{ sorted}$ and $l'$ is a permutation of $a'@b'$, which is a permutation of $a@b$, which is a permutation of $l$.