Some of you have asked again about how your custom comparator should work for the map in the cross-reference program. I'll explain it for you. (I'm too nice :-)

For a map or set to keep items in order, it uses a generalized notion of less-than. If the items you store have a "less-than" operator of some sort, then they can be stored in a map or set. Note, that the type of the items does not have to be equality comparable! This allows maps and sets to be used for many different types, even those that don't have a == operator.

You may wonder how the container knows if you're trying to insert something that's already in there if your objects don't have a == operator. Easy: if !(x < y) && !(y < x), then the items are "equivalent". We say "equivalent" because you can store records in maps where only a part of the record is the logical key. When only a subset of a data item is used for comparison for the ordering, a == operator wouldn't work anyway.

So… a map or set uses the < operator, or a similar operator you provide, to order items and avoid duplicates. To provide your own comparator, you must define a callable with the signature **bool operator( )(const string&, const string&) const**.

If you define your own such operator, it must return **true** if two keys being compared are in order: that is, they are not equivalent keys, and the first precedes the second in your desired ordering.

So how should you do the comparator for the cross-reference program?

Consider what we want. We don't want all the keys beginning with capital letters to come before all the keys that don't, so a simple string comparison won't work. So basically, we want to compare strings ignoring case. This will allow "hello" to precede "The", which it wouldn't if we did a simple comparison based on ASCII values.

So, it sounds like using **_stricmp** (or **strcasecmp** on UNIX) will solve the problem.

Almost. By just doing a case-insensitive comparison, "Hello" and "hello" would be considered equivalent, and therefore both won't be allowed in the map. So, for the special case where the keys are equivalent ignoring case, if we want to keep both keys in the map, we must do a normal, case-sensitive comparison, but *only then*.

It's that simple.