

- c) The key of the master file record will be **greater** than that of the sales file record—that is, we have found a sales file record for which there is **no corresponding master file record**.

In the first case we simply add the quantity sold from the sales file record to the quantity sold in the master record and then fetch another record from each of the lists.

In the second case we simply need to get the next master file record.

In the third case, because the record refers to a book that does not exist in our shop, we copy the information to the error file and then fetch the next sales record.

We then just repeat the above until we have dealt with all records in **both** files.

Testing the keys

How do we test to see whether the records match? Well, just like in a database, we think of each record as having a unique key. In our case it is a combination of the publisher code and ISBN (the files are sorted by ISBN within publisher code). We tend to think of publisher code and ISBN as two separate variables. However, if we convert them into one compound entity, then comparison is straightforward.

For example, instead of reading the two fields separately from the sales file, read them into one single (and larger) string. We can do this using the following declarations:

```
char mbigkey [14], /* Master Compound key - ISBN + Publisher */
    sbigkey [14] ; /* Sales Compound key - ISBN + Publisher */
LINK book_ptr ; /* Pointer to a master record */
```

So, to read a sales record, we simply do the following:

```
fscanf (sales_file, "%s%d", sbigkey, &quantity_sold) ;
```

The master file is already stored in the linked list, so we do not need to use any file operations. Instead, we need to create a new compound key from the existing publisher and ISBN. Assuming that we have already set `book_ptr` to point to a master file record, we can use the following string functions to create the compound key in `mbigkey`.

```
strcpy (mbigkey, book_ptr->publisher) ;
strcat (mbigkey, book_ptr->isbn) ;
```

The first statement copies the `publisher` field into `mbigkey`. The second concatenates (adds) the `isbn` field to the end of `mbigkey`.

You can then compare the two keys, e.g.,
`if (strcmp (mbigkey, sbigkey) == 0).`

Linking your project

There have been problems with getting TLINK to work when trying to run your project. The solution in the readme file works on the 7th floor. In room 321b a different approach is needed to solve the dreaded 'TLINK' error. Assuming you have all your project files stored in a folder called cwk2 on drive M (M:\cwk2), with a project file called `books.ide` the following will work nicely. If you use a different folder and/or project file, then just change the name appropriately in the following.

1. Start up notepad (Start|Run then type notepad and Enter).
2. Type the following three lines into notepad:

```
I:\bc5\bin tlink @books.r$p  
copy m:\books.exe  
books
```
3. Save the file as `m:\cwk2\link.bat` (do this by choosing File|Save As, selecting type "all files" from the drop-down list, use the tree to navigate to `m:\cwk2`, and type the file name enclosed in double quotes "link.bat"; you need the double quotes to stop notepad adding a .TXT extension.
4. Now, every time you want to recompile and run your project: double click on the project node in the C++ compiler as before to create all the OBJ files. You will get the "TLINK" error. Then, from windows explorer, just double click on the file LINK.BAT and this will invoke TLINK, copy the resultant .EXE file back into `m:\cwk2` and run it.

Enjoy.