Chapter 8 Merging and Sorting

Merging: The basic problem is, given two lists, we would like to form another list that is either the intersection or the union of the original lists

Intersection

List 1	List 2	Merged List
Aragorn	Arthur	Arthur
Arthur	Babs	Bilbo
Bilbo	Bilbo	Boramir
Boramir	Boramir	Dirk
Dirk	Dirk	Pippin
Fenchurch	Ellanor	Rose
Ford	Galladrial	Thor
Gandolf	Merry	
Ghimli	Pippin	
Marvin	Rose	
Pippin	Thor	
Rose		
Thor		
Trillium		
Zaphod		

Union

List 1	List 2	Merged List
Aragorn	Arthur	Aragorn
Arthur	Babs	Arthur
Bilbo	Bilbo	Babs
Boramir	Boramir	Bilbo
Dirk	Dirk	Boramir
Fenchurch	Ellanor	Dirk
Ford	Galladrial	Ellanor
Gandolf	Merry	Fenchurch
Ghimli	Pippin	Ford
Marvin	Rose	Galladrial
Pippin	Thor	Gandolf
Rose		Ghimli
Thor		Marvin
Trillium		Merry
Zaphod		Pippin
		Rose
		Thor
		Trillium
		Zaphod

Considerations:

Initialization: start correctly so that the procedure runs smoothly

Getting and Accessing the Next List Item: this should be simple and the main part of the algorithm shouldn't be concerned with it

Synchronization: we shouldn't go past an item when there is a match so where should the next item come from

Handling end-of-file conditions: It shouldn't matter which list we come to the end of first

Error Recognition: If the list is not in order or there are duplicates we should handle these instances elegantly

Intersection: At each step in the processing we examine an element from each list.

`Item_1` is the current item in List 1

`Item_2` is the current item in List 2

What are the possibilities?

- If Item_1 is less than Item_2, we get the next item from List 1
- If Item_1 is greater than Item_2, we get the next item from List 2
- If Item_1 is the same as Item_2, we have a match so we output the item and get the next item from both lists.

The Algorithm: **Match**

Iput : Two sorted lists First_List, and Second_List

Output: The intersection of the two input lists Intersection_List. This list will be sorted also.

Initialize_List connects the list given with the number given

Initialize _Output connects the output list with the list given

Item returns the current item in the indicated list

Next_Item_In_List advances the current item to the next item in the indicated list.

More_Items a boolean value that is true if the call to Next_Item_In_List was able to find an item in the indicated list.

Process_Item this does whatever we want to happen to the item. It might include more than just writing it to a file or putting is into the Result List

Finish_Up when we have found all the matching items we need to close the files and do other house keeping

```
Bool    More_Items;

Initialize_List ( 1,  First_List );
Initialize_List ( 2,  Second_List );
Initialize_Output ( Intersection_List );

More_Items = Next_Item_In_List(1) && Next_Item_In_List(2);

While ( More_Items )
{
    if (  Item( 1 ) < Item ( 2 ) )
        More_Items = Next_Item_In_List( 1 );
    else if (  Item( 1 ) == Item( 2 ) )
    {
        Process_Item( 1 );
        More_Items = Next_Item_In_List( 1 ) &&
        Next_Item_In_List( 2 );
    }
    else
        More_Items = Next_Item_In_List( 2 );
}
Finish_Up( );
```

The Algorithm: **Merge** produces the Union of the two lists

```
Bool  More_Items_1;
Bool  More_Items_2;

Initialize_List( 1, First_List );
Initialize_List( 2, Second_List );

More_Items_1 = Next_Item_In_List( 1 );
More_Items_2 = Next_Item_In_List( 2 );
```

What are the possibilities?

- If Item_1 < Item_2, we take Item_1 and get the next Item from List 1
- If Item_1 > Item_2, we take Item_2 and get the next Item from List 2
- If Item_1 = Item_2, we take either Item and get the next item from both lists.

```
while ( More_Items_1 || More_Items_2 )

    if (  Item( 1 )  <  Item( 2 ) )
    {
        Process_Item( 1 );
        More_Items_1   = Next_Item_In_List( 1 );
    }
    else if ( Item( 1 ) == Item( 2 ) )
    {
        Process_Item( 1 );
        More_Items_1   = Next_Item_In_List( 1 );
        More_Items_2   = Next_Item_In_List( 2 );
    }
    else
    {
        Process_Item( 2 );
        More_Items_2   = Next_Item_In_List( 2 );
    }
    Finish_Up();
```

Assumptions

- Two or more input files are to be processed in a parallel fashion to produce one or more output files
- Each file is sorted on one or more key fields, and all files are ordered in the same ways on the same fields
- In some cases, there must exist a high-key value that is greater than any legitimate record key and a low-key value that is less than any legitimate record key
- Records are to be processed in logical sorted order
- For each file there is only one current record. This is the record whose key is accessible within the main synchronization loop
- Records can be manipulated only in internal memory

Application of the Model to a General Ledger Program

Posting transactions to accounts.

Ledger: A list of Checking and Expense accounts with monthly balances. This list must be sorted by account number.

Journal: For each month, a list of the transactions on each account. Each transaction appears once as an expense and once as a check. This list must be sorted by account number and within each account number by date.

The General Ledger Program Must

- Update the ledger file with the correct balance for each account for the current month
- It must produce a printed version of the ledger that shows
    - the beginning and current balance for each account
    - a list of all the journal transactions for the month

The Algorithm

Item_1 comes from the ledger called the master file

Item_2 comes from the journal called the transaction file

What are the possibilities?

If Item_1 < Item_2,	We have a transaction that does not match the current ledger entry. So, finish the transactions for the current master record and check to see if there is **another master record** and set it up to accept transactions.
If Item_1 > Item_2,	We have a transaction with no master record. This is an error. Get the **next transaction record**.
If Item_1 = Item_2,	We have a transaction for the current master record. Process the transaction and get the **next transaction record**.

So, there are three ways to process the entries

- **Process New Master** Immediately after reading a new ledger object
    - print the header line
    - initialize the balance for the next month from the previous months balance
- **Process Current Master** For each transaction that matches the current master
    - update the account balance
- **Process End Master** After the last transaction for the account
    - print the balance line
    - write the new ledger record to the new ledger file
- **Process Item**
    - print the description of the transaction
- **Process Transaction Error** If there is no ledger account for this transaction
    - print an error message

```
while ( More_Masters || More_Transactions )
{
    if ( Item( 1 )  <  Item( 2 ) )
    {
        Process_End_Master();
        More_Masters  =  Next_Item_In_List( 1 );
        If ( More_Masters ) Process_New_Master();
    }
    else if ( Item( 1 )  ==  Item( 2 ) )
    {
        Process_Current_Master();
        Process_Item( 2 );
        More_Transactions  =  Next_Item_In_List( 2 );
    }
    else
    {
        Process_Transaction_Error();
        More_Transactions  =  Next_Item_In_List( 2 );
    }
}
```

Sample Ledger

Acct No	Account Title	Jan	Feb	Mar	Apr
101	Checking Account #1	1032.57	2114.56	5219.23	
102	Checking Account #2	534.78	3094.17	1321.20	
505	Advertising Expenses	25.00	25.00	25.00	
510	Auto Expenses	195.40	307.92	501.12	
515	Bank Charges	0.00	0.00	0.00	
520	Books and Publications	27.95	27.95	87.40	
525	Interest Expenses	103.50	255.20	380.27	
535	Miscellaneous Expenses	12.45	17.87	23.87	
540	Office Expenses	57.50	105.25	138.37	
545	Postage and Shipping	21.00	27.63	57.45	
550	Rent	500.00	1000.00	1500.00	
555	Supplies	112.00	167.50	2441.80	

Sample Journal Sorted by Account Numbers

Acct. No.	Check No.	Date	Description	Debit/Credit
101	1271	04/02/97	Auto Expense	-78.70
101	1272	04/02/97	Rent	-500.00
101	1273	04/04/97	Advertising	-87.50
101	1274	04/02/97	Auto Expense	-31.83
102	670	04/02/97	Office Expense	-32.78
505	1273	04/04/97	Newspaper ad re: new product	87.50
510	1271	04/02/97	Tune-up and minor repair	78.70
510	1274	04/09/97	Oil change	31.83
540	670	04/02/97	Printer cartridge	32.78
550	1272	04/02/97	Rent for April	500.00

Ledger Print-Out

101 Checking Account #1

1271	04/02/97	Auto Expense	-78.70
1272	04/02/97	Rent	-500.00
1273	04/04/97	Advertising	-87.50
1274	04/02/97	Auto Expense	-31.83

Prev. Bal: 5219.23 New Bal. 4521.20

102 Checking Account #2

670	04/02/97	Office Expense	-32.78

Prev. Bal: 1321.20 New Bal. 1288.42

505 Advertising Expenses

1273	04/04/97	Newspaper ad re: new product	87.50

Prev. Bal: 25.00 New Bal. 112.50

510 Auto Expenses

1271	04/02/97	Tune-up and minor repair	78.70
1274	04/09/97	Oil change	31.83

Prev. Bal: 501.12 New Bal. 611.65

515 Bank Charges

		Prev. Bal:	0.00	New Bal.	0.00	
520	Books and Publications					
		Prev. Bal:	87.40	New Bal.	87.40	
525	Interest Expenses					
		Prev. Bal.	380.27	New Bal.	380.27	
535	Miscellaneous Expenses					
		Prev. Bal.	23.87	New Bal.	23.87	
540	Office Expenses					
	670	04/02/97	Printer cartridge			32.78
		Prev. Bal.	138.37	New Bal.	171.15	
545	Postage and Shipping					
		Prev. Bal.	57.45	New Bal.	57.45	
550	Rent					
	1272	04/02/97	Rent for April			500.00
		Prev. Bal.	1500.00	New Bal.	2000.00	
555	Supplies					
		Prev. Bal.	2441.80	New Bal.	2441.80	