CHAPTER 13

File Structures

(Solutions to Odd-Numbered Problems)

Review Questions

1. The two access methods are sequential and random.
3. The transaction file contains changes that should be made to the old master file.
5. The index is a table that relates the keys of the data items to the addresses in the file where the data are stored.
7. In modulo division hashing, the key is divided by the file size. The remainder plus 1 is used as the address of the record in the file.
9. A collision occurs when two hashed record have the same address. The three collision methods are open addressing, linked list resolution, and bucket hashing. In open addressing, the prime area is searched for an unoccupied address. In linked list resolution, the first record is stored in the home address, but it contains a pointer to the second record. In bucket hashing, a group of records are stored in a buckets which are locations that can accommodate more than one record.

Multiple-Choice Questions

11. d	13. a	15. c	17. a	19. d	21. d
23. a	25. a	27. d			

Exercises

29. The files are shown in Figure S13.29.

Figure S13.29 *Exercise 29*

New Master File

Key	Name	Pay Rate
14	John Wu	17.00
16	George Brown	18.00
17	Duc Lee	11.00
26	Ted White	23.00
31	Joanne King	28.00
89	Mark Black	19.00
90	Orva Gilbert	20.00
92	Betsy Yellow	14.00

Error File

Action	Key	Name	Pay Rate
A	17	Martha Kent	17.00

31.

a. (14232 **mod** 41) + 1 = 5 + 1 = 6

b. (12560 **mod** 41) + 1 = 14 + 1 = 15

c. (13450 **mod** 41) + 1 = 2 + 1 = 3

d. (15341 **mod** 41) + 1 = 7 + 1 = 8

33.

a. 14 + 22 = 36

b. 12 + 57 = 69

c. 13 + 49 = 62

d. 15 + 32 = 47

35.

a. (10278 **mod** 411) + 1 = 3 + 1 = 4

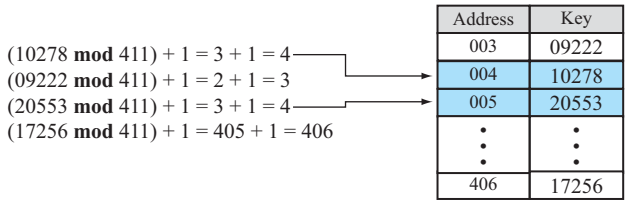
b. (08222 **mod** 411) + 1 = 2 + 1 = 3

c. (20553 **mod** 411) + 1 = 3 + 1 = 4 (collision) → 5

d. (17256 **mod** 411) + 1 = 405 + 1 = 406

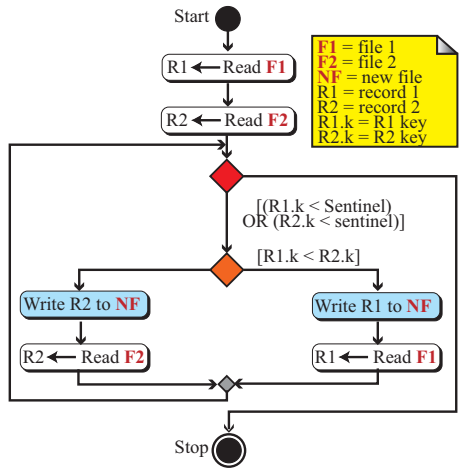
The result of open addressing resolution is shown in Figure S13.35.

Figure S13.35 *Exercise 35*



37. The UML is shown in Figure S13.37. Note that the routine works correctly even if one or both input files are empty (having only one dummy record).

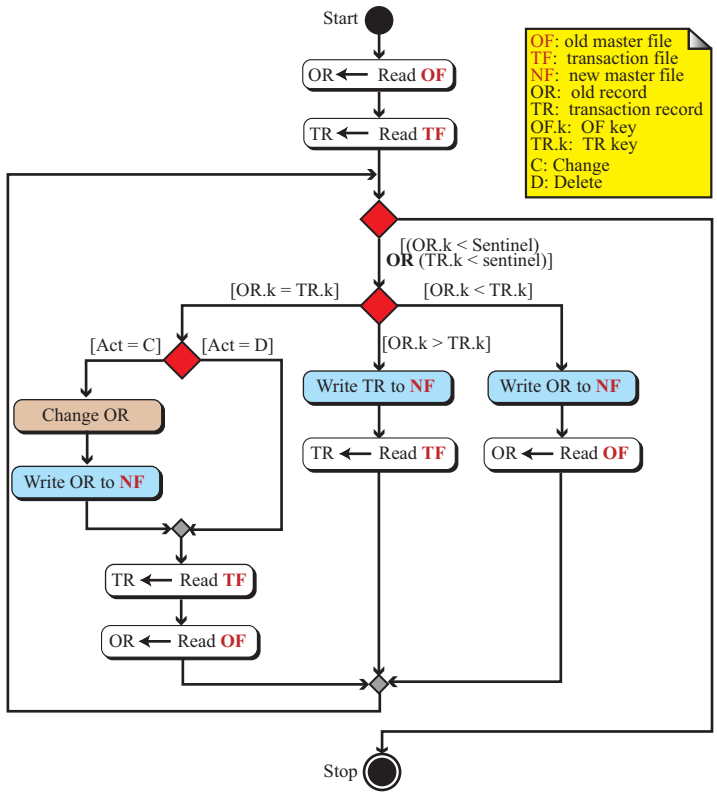
Figure S13.37 *Exercise 37*



39. The UML is shown in Figure S13.41. For simplicity, we assume that there is no discrepancy between OF and TF, which means that there is no need to create an error file. The routine follows the logic in the solution to Exercise 37. The loop will terminate when both files have reached their dummy records. The process, however, needs three decision branches. If the there is no transaction for a record (OR.k < TR.k), the record in the old master file is copied to the new transaction file. If there is a new record in the transaction file to be inserted into the new master file (OR.k > TR.k), then the routine simply writes that record. If (OR.k = TR.k), it means that a record needs to be either partially changed or totally deleted. In the case of deletion, no action is needed. In the case of change, the record in the old master file is updated and written to the new master file. Note that the routine

works correctly if the transaction file is empty (having only one dummy record). This may happen, if at the end of the updating period, there is no changes to the old master file. The old master file is simply copied to the new master file. The routine also works correctly if the old master file is empty (creation of a new file).

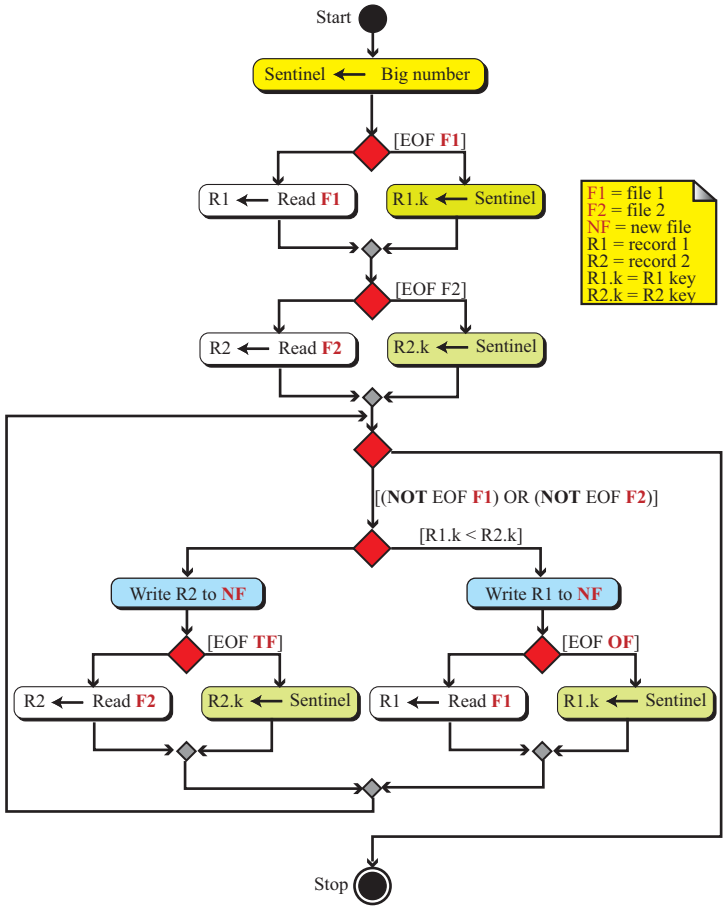
Figure S13.39 *Exercise 39*



41. The UML is shown in Figure S13.41. To simplify the diagram we assume that there is no error. Note that the above digram is similar to the digram shown in Exercise 37. The only difference is that we need to create a dummy record for each file if the file has reached its end. For example, when F1 reaches its end, we store the sentinel value in the key field of the of R1 (a dummy record). So in the next iteration, this record is not selected for processing. When both files reach their ends, the loop is terminated. Note that we first test the status of the end-of-file before reading a record from the file. This is needed because some system creates an error if we try to read from a file when the file has reached its end. Note that the routine works correctly even if one file or both files are empty. If one of the file is

empty, the other file is copied to the new file. If both files are empty, the routine never enters the loop.

Figure S13.41 *Exercise 41*



43. The UML is shown in Figure S13.43. To simplify the diagram we assume there is no error. The diagram combines the logic in Exercises 37, 39, and 41.

Figure S13.43 *Exercise 43*