

Solutions

Each answer shows the steps, then the **result**.

2.1

- a **serial** file keeps records in the **order they were added** (unsorted)
- a **sequential** file keeps them **sorted by a key field**

2.2

- a **random (direct-access)** file

2.3

- it turns a **record key** into an **address/slot** where the record is stored

2.4

- when **two different keys** hash to the **same address/slot**

2.5

- a use: a **data-logging** file, or an unsorted **transaction file**
- reason: records only need to be **added at the end** in the order they arrive; no sorting is needed while they are collected

2.6

- (a) **sequential** – every record is processed in one run
- (b) **direct** – one record is needed quickly
- (c) **direct** – one record is needed quickly

2.7

- $53 \text{ MOD } 10 = 3$; slots 3 and 4 are full, so linear probing moves on to slot **5**

3.1

- $45 \text{ MOD } 10 = 5$
- $23 \text{ MOD } 10 = 3$
- $30 \text{ MOD } 10 = 0$

3.2

- $35 \text{ MOD } 10 = 5$, but slot 5 is **taken** by 45
- linear probing tries the **next slot** (slot 6), and stores 35 there (the first free slot, wrapping if needed)

3.3

- advantage: very **fast direct lookup** by key
- disadvantage: reading in **key order** is harder, and collisions must be handled (space may be wasted)

3.4

- a **random (direct-access)** file using a **hash** of the account number
- because it gives near-instant lookup of a single record without searching the whole file

3.5

any three:

- a **mathematical formula** / calculation
- **applied to the key field** of the record being stored or searched for
- the **result is the address** where the record is stored / should be found
- used for **direct access** to records in a random (or sequential) file

3.6

- **open hashing / linear probing:** store the record in the **next free** location after the one calculated
- **overflow area:** store the record in the **next free space in a separate overflow area**

3.7

(a)

- $24 \text{ MOD } 11 = 2 \rightarrow$ slot **2**
- $13 \text{ MOD } 11 = 2 \rightarrow$ 2 full $\rightarrow$ slot **3**
- $35 \text{ MOD } 11 = 2 \rightarrow$ 2, 3 full $\rightarrow$ slot **4**
- $21 \text{ MOD } 11 = 10 \rightarrow$ slot **10**; $46 \text{ MOD } 11 = 2 \rightarrow$ 2, 3, 4 full $\rightarrow$ slot **5**

(b)

- slots 2, 3, 4, 5 are compared: **4** comparisons

(c)

- $57 \text{ MOD } 11 = 2$; slots 2, 3, 4, 5 hold other keys and slot **6 is empty**
- an empty slot ends the search, because 57 would have been stored there: it is **not in the table**

3.8

one mark for each completed gap (max 6)

- OPENFILE "Members.dat" FOR RANDOM
- WHILE NOT Found AND Location <= 5000
- SEEK "Members.dat", Location
- GETRECORD "Members.dat", ThisID
- IF ThisID = SearchID THEN
- RETURN -1