1.4 DATA TYPES & STRUCTURES · 1.4.2

Hash tables — Mark scheme

30 marks · spec 1.4.2(b)(c)

AO key: AO1 = knowledge · AO2 = application · AO3 = reasoned judgements. Award method marks for correct hashing even with a transcription slip. Do not award the same point twice.

Q	ANSWER	AO	MARKS
1(a)	It calculates an address/index from the key (1); that determines where the item is stored / found in the table (1). (2)	AO1	2
1(b)	The address is calculated directly from the key (1); so the item can be accessed without searching through all the data (near-constant time) (1). (2)	AO1	2

Q	ANSWER	AO	MARKS
2(a)	<table><tr><td>KEY</td><td>41</td><td>23</td><td>67</td><td>33</td><td>13</td></tr><tr><td>MOD 10</td><td>1</td><td>3</td><td>7</td><td>3</td><td>3</td></tr><tr><td>STORED</td><td>1</td><td>3</td><td>7</td><td>4</td><td>5</td></tr></table> Correct hashes (1); 41→1, 23→3, 67→7 placed (1); 33 collides at 3 → 4 (1); 13 collides at 3 (and 4) → 5 (1); all final addresses correct (1). (5)	AO2	5
2(b)	33 (or 13) hashed to address 3, already holding 23 — a collision (1); it was placed in the next free cell (4 for 33; 5 for 13) (1). (2)	AO2	2
2(c)	Hash 33 to 3, find a different key, check successive cells (4) until 33 is found. (1)	AO2	1

Q	ANSWER	AO	MARKS
3(a)	When two different keys hash to the same address. (1)	AO1	1
3(b)	Chaining: each table cell holds a list (linked list) (1); colliding items are added to the list at that address (1). (2)	AO1	2
3(c)	A search checks successive cells until it reaches a blank/empty cell, taking that to mean "not present" (1); an emptied cell looks blank, so the search stops early and misses an item stored beyond it (1); prevented by placing a "deleted" marker (tombstone) so searches continue past it (1). (3)	AO2	3

Q	ANSWER	AO	MARKS
4(a)	How full the table is (1); = number of items stored ÷ table size (1). (2)	AO1	2
4(b)	150 ÷ 200 = 0.75 (75%). (1)	AO2	1
4(c)	More cells are occupied so collisions become more frequent (1); probing has to check more cells, so adding/searching is slower (1). (2)	AO2	2

Q	LEVELS-OF-RESPONSE MARK SCHEME	AO	MARKS
5	Mark using the levels descriptors below. AO1 (hash table/array knowledge), AO2 (application to the lookup), AO3 (justified judgement).	AO1 ×2 AO2 ×3 AO3 ×2	7

LEVEL	DESCRIPTOR
Level 3 (6–7)	Accurately compares hash table and sorted array for fast ID lookup, referring to speed, collisions and load factor, clearly **applied**, with a **justified** recommendation. Well structured.
Level 2 (3–5)	Covers both with **some** application and partial justification.
Level 1 (1–2)	Basic points about one or both structures with little application.
0	Nothing creditworthy.

Indicative content (credit any valid point):

- Hash table gives near-constant-time lookup by calculating the address from the ID — ideal for fast single-record lookup.
- Binary search on a sorted array is O(log n) — fast but slower than a well-tuned hash table, and the array must be kept sorted (costly inserts).
- Hash table downsides: collisions slow it down; a high load factor degrades performance, so it must be sized/resized appropriately.
- Hash tables don't keep data in order (poor for range queries); the sorted array does.
- Justified recommendation: a hash table suits fast lookup by exact ID, provided the load factor is kept low.

Total for paper: 30 marks