2.5 Data Structures and Their Operations: 10-minute Pack

Total Marks

OCR H446 · Calculators not allowed · about 10 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 A print-queue manager used in a school office in Norwich implements a linear queue, a data structure that operates on a First In First Out (FIFO) basis, using an array of size 5 (indices 0 to 4). Two pointers are used: front (the index of the item at the front of the queue) and rear (the index of the item most recently added). Initially front = 0 and rear = -1 (the queue is empty).

(a) State what is meant by the term 'First In First Out (FIFO)'. **(1)**

(b) The following operations are performed, in order, on the initially empty queue: enqueue(A), enqueue(B), enqueue(C), dequeue(), enqueue(D) **(5)**

Complete a trace table showing the array contents, front and rear after each operation.

(c) Explain one disadvantage of a linear queue, compared with a circular queue, in the context of the array above once index 0 has been vacated by the dequeue in part (b). **(2)**

(d) Describe how a circular queue would allow index 0 to be reused once rear reaches the last index of the array. **(2)**

(Total for Question 1 is 10 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.5 Data Structures and Their Operations: 10-minute Pack

Question 1

(a) the item that has been in the structure the longest (added first) is the first one to be removed (oe) **B1**

(a) Answer: The item that was added earliest is the first one to be removed.

(b) enqueue(A): array = [A,_,_,_,_], front = 0, rear = 0 **B1**

(b) enqueue(B): array = [A,B,_,_,_], front = 0, rear = 1 **B1**

(b) enqueue(C): array = [A,B,C,_,_], front = 0, rear = 2 **B1**

(b) dequeue(): removes A, front = 1, rear = 2 **B1**

(b) enqueue(D): array = [_,B,C,D,_], front = 1, rear = 3 **B1**

(b) Answer: After all five operations: front = 1, rear = 3, and the queue (front to rear) holds B, C, D.

(c) once an item is dequeued from the front, that array slot (e.g. index 0) becomes unusable in a linear queue, since only rear moves forward on enqueue (oe) **B1**

(c) as more items are enqueued and dequeued, rear keeps advancing towards the last index, so the queue can eventually report as full (no more room after rear) even though earlier slots are actually free, wasting space (oe) **B1**

(c) Answer: Index 0 becomes wasted space once A is dequeued, because a linear queue only ever moves rear forward; eventually rear reaches the final index and the queue is reported full, even though slots such as index 0 are actually empty.

(d) the rear pointer wraps around back to index 0 using modulo arithmetic, e.g. rear = (rear + 1) MOD size, once it passes the last index (oe) **B1**

(d) this reuses slots that have already been dequeued (are now outside the front-to-rear range), as long as the queue is not already full, so space is not wasted (oe) **B1**

(d) Answer: In a circular queue, rear is calculated using rear = (rear + 1) MOD size, so once rear passes the last index it wraps back to index 0; provided that slot has already been freed by an earlier dequeue, it can safely be reused instead of being left wasted.

© Revision Library | revisionlibrary.co.uk