Q.6: Explain any one of the data structures in Q.5, with an example from your day to day life. Discuss how it can be implemented when you try to make a computational simulation of the same.

Ans:6 :→ Queue Data Structure :→

As discussed in the previous question, a queue data structure is based on FIFO (first-in-first-out) principle. A real life example that I can think of is of the 'print queue', when different computers connected to the same printer request print. The requests are added to the "print queue" and are implemented on the basis of first-in-first-out principle, where the user who requests print before some other user gets his file printed before the latter. Clearly, the task needs to be performed in a sequential manner, one at a time. Moreover, the print queue must run until the queue becomes empty, i.e until all the requests are processed.

To implement a print queue, we have two options at our disposal - using arrays or using linked lists. I think that for the case of print queue, dynamic memory allocation will be more useful, since a limitation of memory imposed by arrays may turn to be a liability. So I prefer to use the linked list implementation of "queues".

Algorithm :→

Step : ① Declare the structure for queue-elements. Each queue element contains two fields, one for storing the value and another for storing a pointer to the next element.

Step : ② Declare two pointers front and rear to the structure and initialize them to the Null value.

Inserting an element in the queue :→ When a print request is placed, a new queue element must be added to the rear end. This is done using a pointer ptr that uses the malloc function to allocate memory required for the queue structure. I think the pointer element value will consist of the file to be printed and it will contain the address of Null.

Step : ③ : Declare a pointer ptr to the queue structure data type and dynamically allocate memory to it.

Step:④ :→ The ptr value is made equal to the file value. The address must be set to null.

[If this the first value in the queue, then set front = rear = ptr and then ptr → Next = Null]

[If this is added to a non-empty queue, then set

1. rear → Next = ptr , 2. ptr → Next = Null , 3. rear = ptr so that ptr becomes the new rear element].

This step is continued untill the requests are made.

Removing the printed file : once the file pointed by front is printed, it has to be removed from the queue.

Step:⑤ :→ Declare an integer 'i' :→ int i .

Step:⑥ :→ Initialize 'i' to the value of the front pointer and set front = front → next.

Continue steps ⑤ and ⑥ untill the front pointer points to NULL.

Step:⑦ :→ Continue till front == NULL

Step:⑧ :→ Exit the queue.

Thus, in the above ways, one may execute a 'print' queue using linked lists.