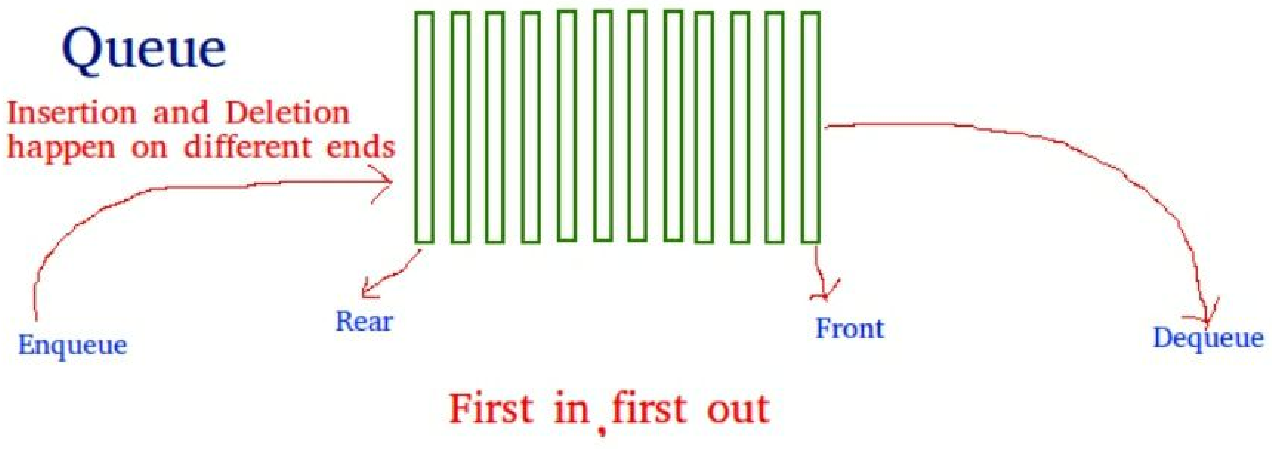
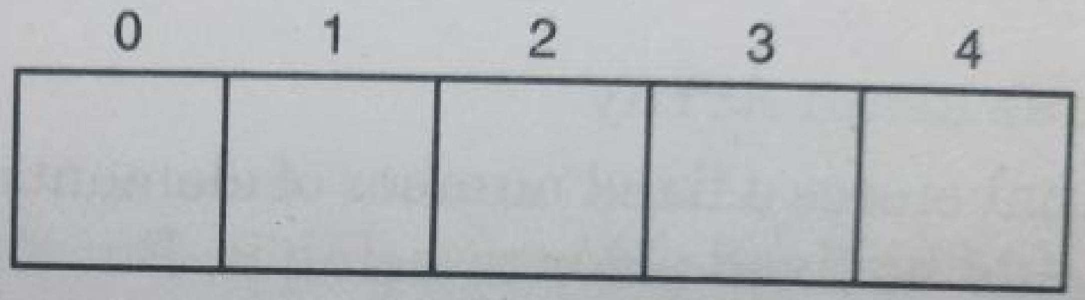
A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.



**Example :**

Suppose we have an empty queue, with 5 memory cells such as :

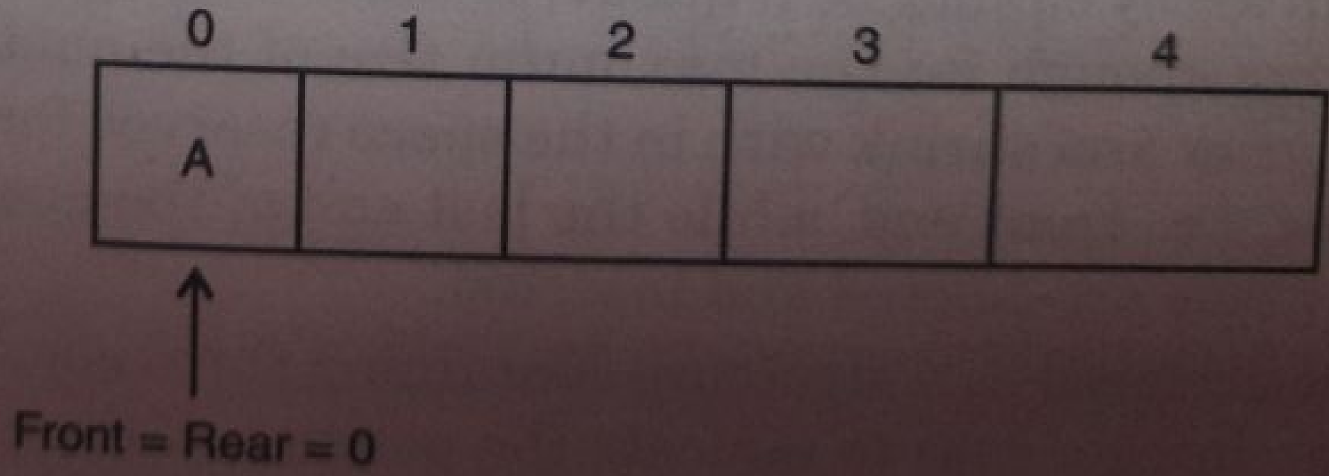


Now Front = – 1

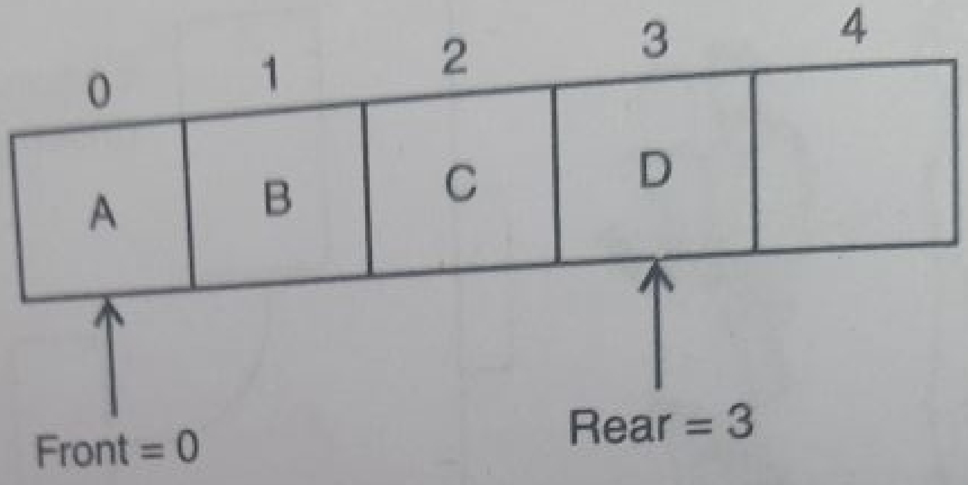
Rear = – 1 *i.e.*, Empty queue.

**Insert A :**

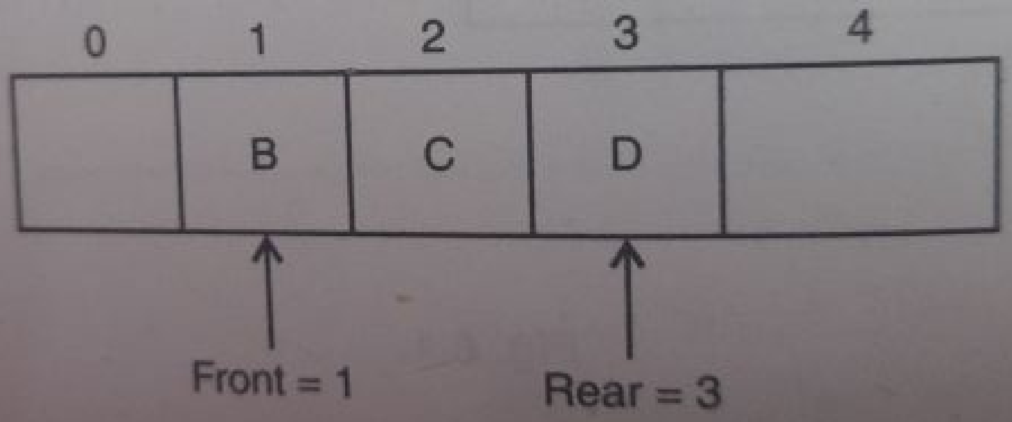
Now first element A is inserted in queue.



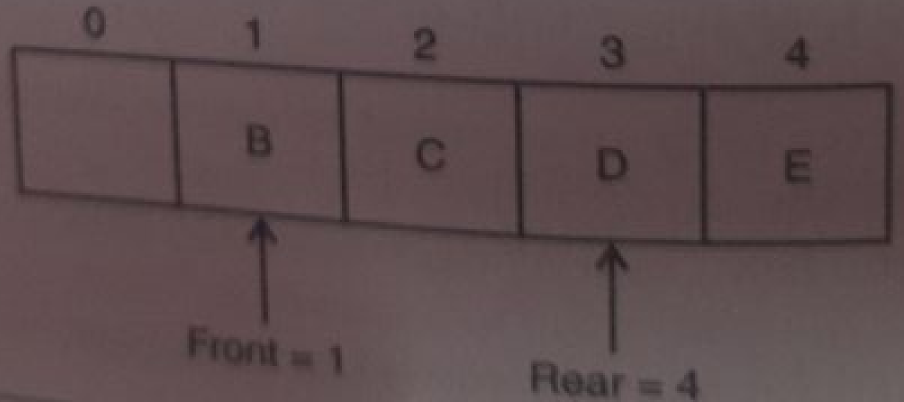
Insert B, C, D :



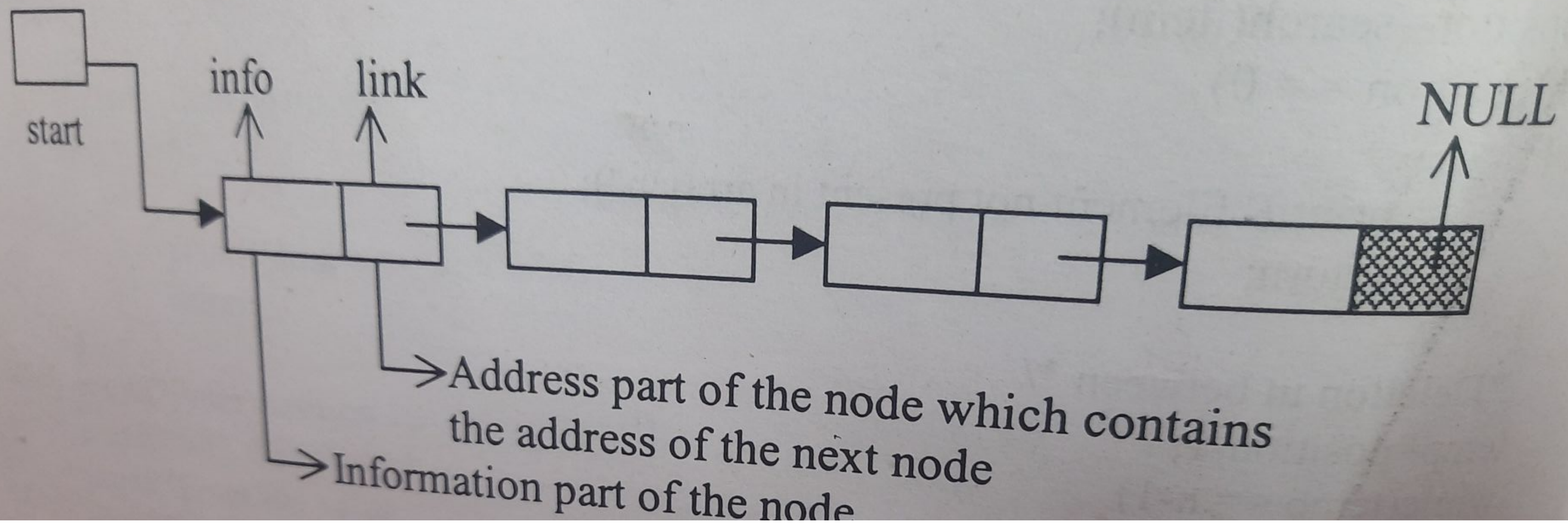
Deletion of an Element :



Insertion of E :



§ 6.2 REPRESENT


start
info
link
NULL
Address part of the node which contains the address of the next node
Information part of the node

A linked list is an ordered collection of finite homogeneous data elements called nodes where the linear order is maintained by means of links or pointers. That is, each node is divided into two parts : the first part contains the information of the element and the second part, called the link field or the next pointer field, contains the address of the next node in the list.

The number of pointers is maintained depending on the requirements and usage. Based on this, linked list are classified into three categories :

1. Linear linked list
2. Doubly linked list
3. Circular linked list

**Linear Linked List :** In linear linked list, each node is divided in two parts : First part contains the information of the element. And the second part contains the address of the next node in the list. That is, each node has a single pointer to the next node, and in the last node's case a null pointer representing that there are no more nodes in the linked list.

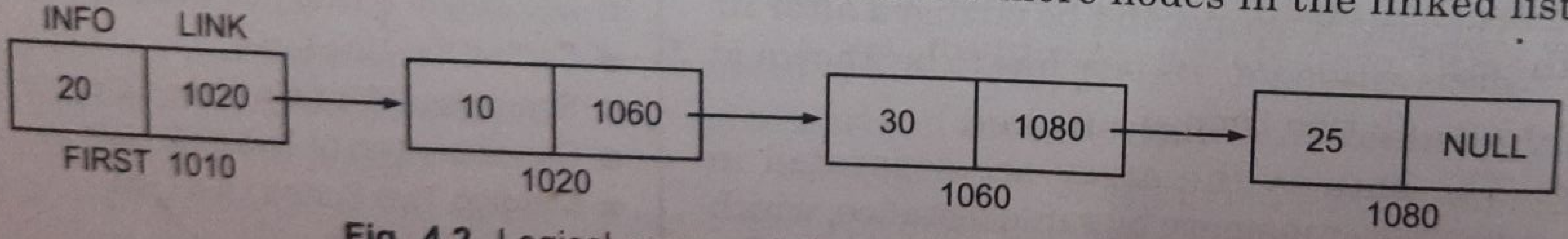


**Fig. 4.2.** Logical representation of singly linked list

**Circular Linked List :** It is just like a linear linked list in which the second part of the last node contains the address of the first node in the list *i.e.*, the second field of the last node does not point to the null pointer rather it points back to the beginning of the linked list.

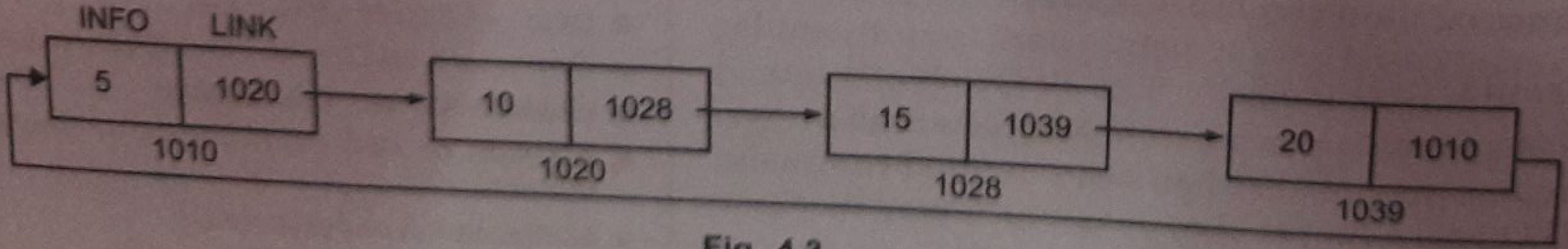


**Fig. 4.3.**

**Doubly Linked List :** Doubly linked lists are also called two-way lists. In the case of doubly linked list, two link fields are maintained instead of one as in singly linked list which help in accessing both the successor and predecessor nodes. A structure of the doubly linked list is shown in figure.

Fig. 4.2. Logical representation of singly linked list

**Circular Linked List :** It is just like a linear linked list in which the second part of t
last node contains the address of the first node in the list *i.e.*, the second field of the la
node does not point to the null pointer rather it points back to the beginning of the linke
list.

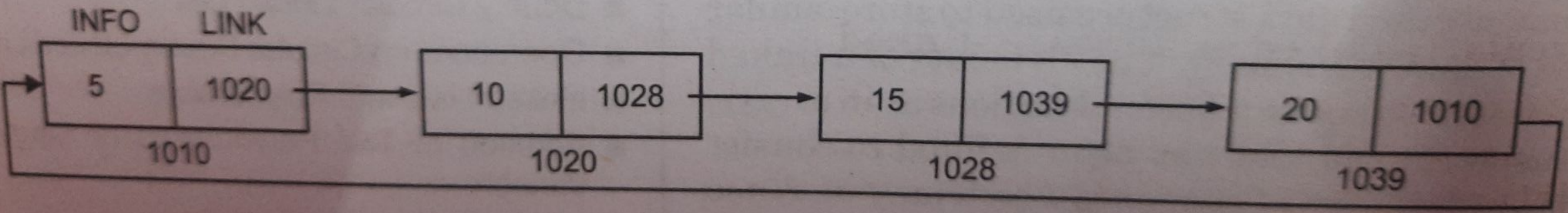
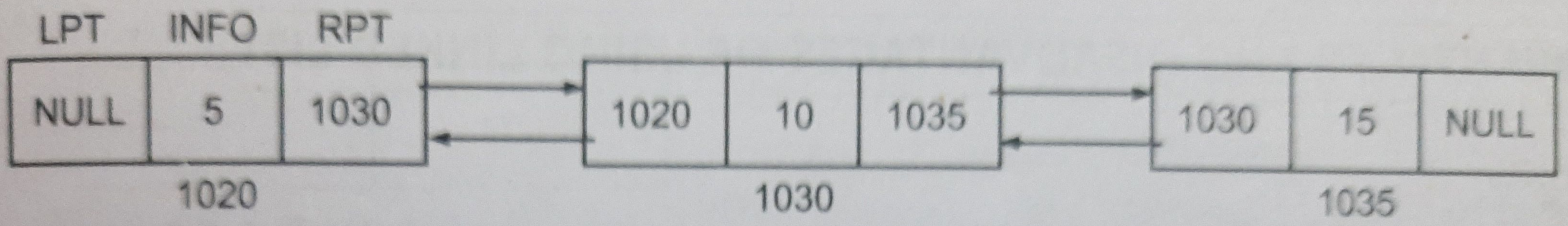


Fig. 4.3.

**Doubly Linked List :** Doubly linked lists are also called two-way lists. In the case of doubly linked list, two link fields are maintained instead of one as in singly linked list which help in accessing both the successor and predecessor nodes. A structure of the doubly linked list is shown in figure.


LPT
INFO
RPT
NULL
5
1030
1020
1020
10
1035
1030
1030
15
NULL
1035