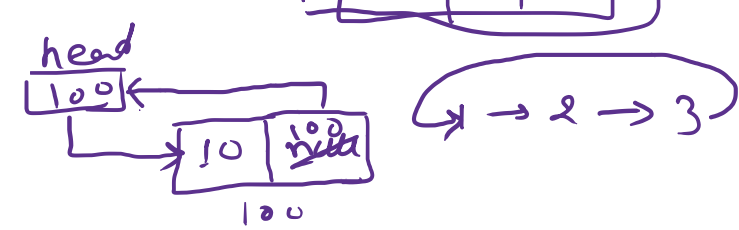


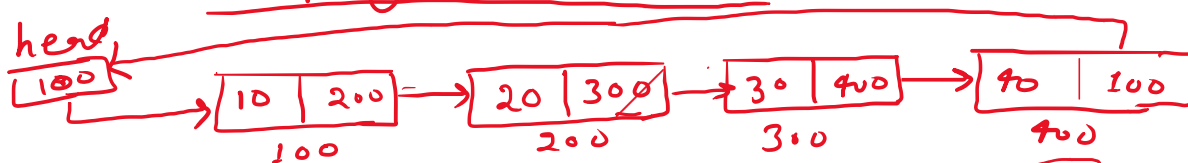
- ① Bubble Sort
- ② Insertion Sort
- ③ Selection sort
- ④ merge. sort
- ⑤ heap sort
- ⑥ Quick sort

- ① ~~ins~~ Create list and display list
- ② insert beginning, end, between
- ③ Delete beginning, end, between.
- ④ Reverse Circular linked list

Diagram illustrating a linked list structure with 5 nodes. The head points to the first node (10, 200). The next pointers are 20, 30, 40, and 50. The last node's next pointer is null. The diagram shows the process of inserting a new node (50, 100) at the end. The new node is added, and the previous node's next pointer is updated to point to it. The diagram also shows the deletion of a node (40, 400) by bypassing it. The diagram includes labels for 'Node', 'int data', 'Node next', and 'Node prev'.

$$C_1 = 100$$


Display corridor List



output = 10 20 30 40

$CI = \text{Current}$

$c = \overset{100}{\cancel{1000}} \overset{100}{\cancel{200}} \overset{100}{\cancel{400}}$

$\text{while } (c \overset{100}{=} \overset{100}{\text{head}}) \{$

$\boxed{100 \overset{100}{=} 100}$

$100 \rightarrow 700$

(T) - $\frac{200!}{2} = 100$

The diagram illustrates a linked list with a cycle. The nodes are represented as boxes containing their value and their next pointer. The sequence of nodes is: 10 (points to 20), 20 (points to 30), 30 (points to 40), and 40 (points back to 10). A separate node with value 50 is shown with a null pointer (indicated by 'null' and a crossed-out '00'). A red arrow labeled '50' points to the start of the list (node 10). A blue arrow labeled 'X' points from the null pointer of node 50 to the start of the list, indicating the point where the cycle is detected.

```

n1.next = head;
head = n1;
cr.next = head;

```

16-07-2023 covered topic

- ① create circular list
- ② display circular list
- ③ insert beginning of the list

17-07-2003 to PIC

- ① insert last of the circular list
- ② insert between of the circular list
- ③ delete operation (beginning, last, between)