**WEEK 7**

**AIM: To perform operations on different types of linked list**

In this exercise, I implemented three different types of linked lists i.e singly linked list, doubly linked list, and circular linked list. I have given the user 4 operations to perform on each linked list type

1. Insertion
2. Deletion
3. Search
4. Print

**VARIABLES USED**

Node of linked list: node

Data in node: data

Link to next node: next

Link to previous node: prev

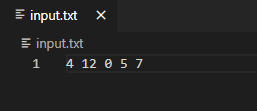
Element to search: elem

Position of the number being searched: pos

File pointer: fptr

Helper variable: choice,temp , curr, i

Initial content of input file:



All four operations are explained below in detail:

1. **Insertion:** This function is used to insert in the linked list in sorted orders so that nodes are connected in ascending order of their data value.

**Singly Linked List**

Step 1) If the linked list is empty then make the node as head, set the next pointer to NULL, and return it.

Step 2) If the value of the node to be inserted is smaller than the value of the head node, then insert the node at the start and make it head.

Step 3) In a loop, find the appropriate node after which the input node is to be inserted. To find the appropriate node start from the head, keep moving until you reach a node whose value is greater than the input node. The node just before the greater node is the appropriate node.

Step 4) Insert the node after the appropriate node found in step 3 and set the next pointers accordingly.

**Doubly Linked List**

Step 1) If the linked list is empty then make the node as head, set next and the previous pointer to NULL, and return it.

Step 2) If the value of the node to be inserted is smaller than the value of the head node, then insert the node at the start, set the next pointer to head and the previous pointer to NULL, and make it head.

Step 3) In a loop, find the appropriate node after which the input node is to be inserted. To find the appropriate node start from the head, keep moving until you reach a node whose value is greater than the input node. The node just before the greater node is the appropriate node.

Step 4) Insert the node after the appropriate node found in step 3 and set pointers accordingly.

**Circular Linked List**

Step 1) If the linked list is empty then make the node as head, set next pointer to itself, and return it.

Step 2) If the value of the node to be inserted is smaller than the value of the head node, then insert the node at the start, set the next pointer to head and the next pointer of the head to this new node, and make it head.

Step 3) In a loop, find the appropriate node after which the input node is to be inserted. To find the appropriate node start from the head, keep moving until you reach a node whose value is greater than the input node. The node just before the greater node is the appropriate node.

Step 4) Insert the node after the appropriate node found in step 3 and set pointers accordingly.

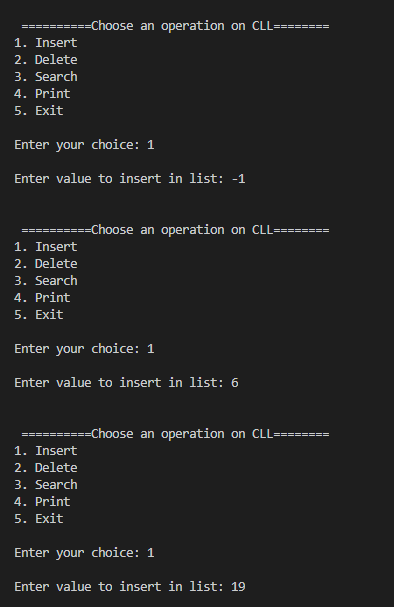
Time complexity: Best case: O(1)

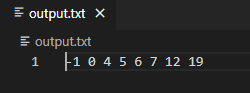
Average Case: O(n)

Worst Case: O(n)

Sample Input from file: 4 12 0 5 7

User Input: -1 6 19





1. **Deletion:** This function is used to remove a node from the linked list based on the data value.

**Singly Linked List**

Step 1) If the linked list is empty then print List is empty and return NULL.

Step 2) If .the searched value is head then delete head and set head to next node (set to NULL if there is nothing).

Step 3) In a loop, search the required node sequentially. If the node to be deleted is found, connect its previous node to next node and delete the node.

Step 4) If the node is not found then print a message that node not found and simply return.

**Doubly Linked List**

Step 1) If the linked list is empty then print List is empty and return NULL.

Step 2) If .the searched value is head then delete head and set head to next node (set to NULL if there is nothing).

Step 3) In a loop, search the required node sequentially. If the node to be deleted is found, connect its previous node and the next node accordingly and delete the node.

Step 4) If the node is not found then print a message that node not found and simply return.

**Circular Linked List**

Step 1) If the linked list is empty then print List is empty and return NULL.

Step 2) If .the searched value is head then delete head and set head to next node (set to NULL if there is nothing).

Step 3) In a loop, search the required node sequentially. If the node to be deleted is found, connect its previous node to the next node and delete the node.

Step 4) If the node is not found then print a message that node not found and simply return.

Time complexity: Best case: O(1)

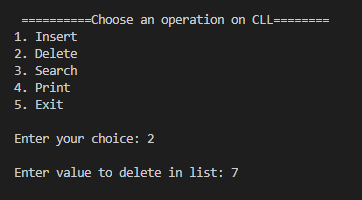
Average Case: O(n)

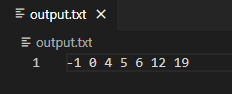
Worst Case: O(n)

Current linked list: -1 0 4 5 6 7 12 19

Input: 7

Output: -1 0 4 5 6 12 19





1. **Search:** This function is used to search a node having the required data value and return its position in the linked list.

Search technique is same in given 3 versions of linked list

Step 1) If the linked list is empty then print List is empty and return -1.

Step 2) In for loop, sequentially compare every node data value if it matches then return loop counter value as position.

( termination condition of the loop in singly linked list and the doubly linked list is a NULL node and in case of circular linked list is head itself)

Step 3) If after complete traversing node is not found then return -1.

Time complexity: Best case: O(1)

Average Case: O(n)

Worst Case: O(n)

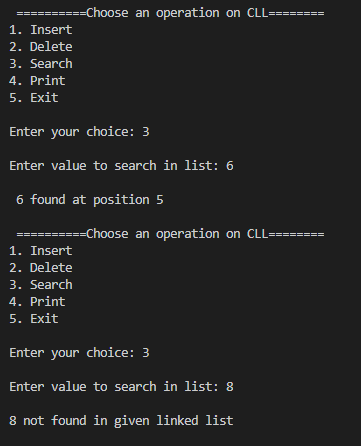
Current linked list: -1 0 4 5 6 7 12 19

Input: 6

Output: 6 found at position 5

Input: 8

Output: 8 not found in the list



1. **Print:** This function is used to traverse the whole linked list and print data values of the nodes.

Traversal technique is same in given 3 versions of linked list

Step 1) If the linked list is empty then print List is empty and return.

Step 2) In for loop, sequentially print every node data value and advance the current pointer to the next node.

( termination condition of the loop in singly linked list and the doubly linked list is a NULL node and in case of circular linked list is head itself)

Time complexity: O(n)

Sample Input from file: -1 0 4 5 6 12 19

