NAME:SAYALI JIVAN CHAUDHARI

ROLL NO.:14

PRN NO.2023015400005055

1)implementation of doubly linked list

```
#include <iostream>
using namespace std;
struct Node {
  int data;
  struct Node *prev;
  struct Node *next;
};
struct Node* head = NULL;
void insert(int newdata) {
  struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
  newnode->data = newdata;
  newnode->prev = NULL;
  newnode->next = head;
  if(head != NULL)
  head->prev = newnode ;
  head = newnode;
```

```
}
void display() {
  struct Node* ptr;
  ptr = head;
  while(ptr != NULL) {
    cout<< ptr->data <<" ";
    ptr = ptr->next;
  }
}
int main() {
  insert(3);
  insert(1);
  insert(7);
  insert(2);
  insert(9);
  cout<<"The doubly linked list is: ";
  display();
  return 0;
}
// C++ program to delete a node at any position in
// Doubly Linked List
#include <bits/stdc++>
```

```
#include<iostream.h>
using namespace std;

// Anode of the doubly linked list
class Node
{
	public:
	int data;
	Node* next;
	Node* prev;
};

/* Function to delete a node in a Doubly
Linked List. head_ref --> pointer to
head node pointer. del --> pointer to
node to be deleted. */
void deleteNode(Node** head_ref, Node* del)
{
	// Base case
	if (*head_ref == NULL || del == NULL)
		return;
```

```
        // If node to be deleted is head node
        if (*head_ref == del)
                *head_ref = del->next;

        /* Change next only if node to be
        deleted is NOT the last node */
        if (del->next != NULL)
                del->next->prev = del->prev;

        /* Change prev only if node to be
        deleted is NOT the first node */
        if (del->prev != NULL)
                del->prev->next = del->next;

        /* Finally, free the memory occupied
        by del*/
        free(del);
        return;
}

// UTILITY FUNCTIONS
/* Function to insert a node at the
```

```
beginning of the Doubly Linked List */
void push(Node** head_ref, int new_data)
{
        // Allocate node
        Node* new_node = new Node();

        // Put in the data
        new_node->data = new_data;

        /* Since we are adding at the
        beginning, prev is always NULL */
        new_node->prev = NULL;

        /* Link the old list off the
        new node */
        new_node->next = (*head_ref);

        /* Change prev of head node to
        new node */
        if ((*head_ref) != NULL)
                (*head_ref)->prev = new_node;
```

```
        /* Move the head to point to the
        new node */
        (*head_ref) = new_node;
}

/* Function to print nodes in a given
doubly linked list. This function is
same as printList() of singly linked list */
void printList(Node* node)
{
        while (node != NULL)
        {
                cout << node->data << " ";
                node = node->next;
        }
}

// Driver code
int main()
{
        // Start with the empty list
        Node* head = NULL;
```

```
/* Let us create the doubly linked list
10<->8<->4<->2 */
push(&head, 2);
push(&head, 4);
push(&head, 8);
push(&head, 10);

cout << "Original Linked list ";
printList(head);

/* Delete nodes from the doubly
linked list */
// Delete first node
deleteNode(&head, head);

// Delete middle node
deleteNode(&head, head->next);

// Delete last node
deleteNode(&head, head->next);
```

```
        /* Modified linked list will be
        NULL<-8->NULL */
        cout << "Modified Linked list ";
        printList(head);
        return 0;
}
```