Program 9 (Doubly LL)

```
typedef struct Node {
        int data;
        struct Node* prev;
        struct Node* next;
} node;

void insertAtEnd(node** head, int d){
    node* n, *temp = *head;
    if (*head == NULL) {
        *head = (node*) malloc (sizeof(node));
        (*head)→ prev = NULL
        (*head) → next = NULL
        (*head) → data = d
    }
    else {
        while(temp→ next != NULL) {
            temp = temp→ next;
        }
        n = (node*) malloc (sizeof(node));
        n→ data = d;
        n→ next = NULL;
        n→ prev = temp;
        temp→ next = n;
    }
}
```

```
void insertAtLeft(node **h, int d, int ele)
    node *head = *h;
    if (head->data == ele){
        node * temp1 = NULL;
        temp1 = (node *)malloc (sizeof(node
        temp1->prev = NULL;
        temp1->data = d;
        temp1->next = *h;
        (*h)->prev = temp1;
        (*h) = temp1;
        return;
    }
    node *temp = NULL;
    while(head != NULL){
        if(head->data == ele){
            head = head->prev;
            temp = (node *)malloc(sizeof(node
            temp->data = d;
            temp->prev = head;
            temp->next = head->next;
            temp->next->prev = temp;
            head->next = temp;
            break;
        }
        else {
            head = head->next;
        }
    }
}
```

```
void deleteNode(node **head, int d) {
    node *temp = *head;
    if (*head == NULL) {
        printf("No elements in
list to delete\n");
        return;
    }
    while(temp != NULL) {
        if (temp -> data == d) {
if(temp == *head) { *head = (*head) -> next;
            *head -> prev = NULL;
        }
        else if (temp -> next == NULL) {
            temp -> prev -> next = NULL;
            free(temp);
        }
        else {
            temp -> prev -> next = temp -> next;
            temp -> next -> prev = temp -> prev;
            free(temp);
        }
        return;
    }
    temp = temp -> next;
  }
}
```