LAB ASSIGNMENT 4

AMISH PRIYADARSHI

102269002

1). #include <stdio.h>

```
#include <stdlib.h>


// Define a structure for a singly linked list node
struct Node {
	int data;
	struct Node* next;
};


// Function to create a new node with the given data
struct Node* createNode(int data) {
	struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
	newNode->data = data;
	newNode->next = NULL;
```

```
    return newNode;
}

// Function to insert a node at the beginning of the linked list
void insertAtFirst(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}

// Function to insert a node at the end of the linked list
void insertAtLast(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
```

```
}

// Function to delete the first node from the linked list
void deleteAtFirst(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
    struct Node* temp = *head;
    *head = (*head)->next;
    free(temp);
}

// Function to delete the last node from the linked list
void deleteAtLast(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
    if ((*head)->next == NULL) {
        free(*head);
        *head = NULL;
```

```
        return;
    }
    struct Node* current = *head;
    while (current->next->next != NULL) {
        current = current->next;
    }
    free(current->next);
    current->next = NULL;
}

// Function to traverse and print the linked list
void traverse(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;
```

```
// Insert elements at the beginning
insertAtFirst(&head, 3);
insertAtFirst(&head, 2);
insertAtFirst(&head, 1);

// Insert elements at the end
insertAtLast(&head, 4);
insertAtLast(&head, 5);

// Print the linked list
printf("Linked List: ");
traverse(head);

// Delete elements from the beginning
deleteAtFirst(&head);
deleteAtFirst(&head);

// Print the updated linked list
printf("Linked List after deletion at the first: ");
traverse(head);
```

```
    // Delete elements from the end
    deleteAtLast(&head);
    deleteAtLast(&head);

    // Print the final linked list
    printf("Linked List after deletion at the last: ");
    traverse(head);

    return 0;
}
```

2).

```
#include <stdio.h>
#include <stdlib.h>

// Define a structure for a doubly linked list node
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
```

```
// Function to create a new node with the given data
struct Node* createNode(int data) {
	struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
	newNode->data = data;
	newNode->prev = NULL;
	newNode->next = NULL;
	return newNode;
}

// Function to insert a node at the beginning of the linked list
void insertAtFirst(struct Node** head, int data) {
	struct Node* newNode = createNode(data);
	if (*head == NULL) {
		*head = newNode;
		return;
	}
	newNode->next = *head;
	(*head)->prev = newNode;
	*head = newNode;
}

// Function to insert a node at the end of the linked list
```

```
void insertAtLast(struct Node** head, int data) {
	struct Node* newNode = createNode(data);
	if (*head == NULL) {
		*head = newNode;
		return;
	}
	struct Node* current = *head;
	while (current->next != NULL) {
		current = current->next;
	}
	current->next = newNode;
	newNode->prev = current;
}

// Function to delete the first node from the linked list
void deleteAtFirst(struct Node** head) {
	if (*head == NULL) {
		printf("List is empty. Cannot delete.\n");
		return;
	}
	struct Node* temp = *head;
	*head = (*head)->next;
```

```
    if (*head != NULL) {
        (*head)->prev = NULL;
    }
    free(temp);
}

// Function to delete the last node from the linked list
void deleteAtLast(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    if (current->prev != NULL) {
        current->prev->next = NULL;
    } else {
        *head = NULL;
    }
    free(current);
```

```
}

// Function to traverse and print the linked list
void traverse(struct Node* head) {
    struct Node* current = head;
    printf("Forward: ");
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");

    printf("Backward: ");
    current = head;
    while (current->next != NULL) {
        current = current->next;
    }
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->prev;
    }
    printf("NULL\n");
```

```
}

int main() {
    struct Node* head = NULL;

    // Insert elements at the beginning
    insertAtFirst(&head, 3);
    insertAtFirst(&head, 2);
    insertAtFirst(&head, 1);

    // Insert elements at the end
    insertAtLast(&head, 4);
    insertAtLast(&head, 5);

    // Print the linked list
    printf("Linked List:\n");
    traverse(head);

    // Delete elements from the beginning
    deleteAtFirst(&head);
    deleteAtFirst(&head);
```

```
    // Print the updated linked list
    printf("Linked List after deletion at the first:\n");
    traverse(head);

    // Delete elements from the end
    deleteAtLast(&head);
    deleteAtLast(&head);

    // Print the final linked list
    printf("Linked List after deletion at the last:\n");
    traverse(head);

    return 0;
}
```

3).

```
#include <stdio.h>
#include <stdlib.h>

// Define a structure for a circular doubly linked list node
struct Node {
```

```
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to create a new node with the given data
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a node at the beginning of the circular doubly linked list
void insertAtFirst(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        newNode->prev = newNode;
        newNode->next = newNode;
    } else {
```

```
        struct Node* last = (*head)->prev;
        newNode->next = *head;
        newNode->prev = last;
        (*head)->prev = newNode;
        last->next = newNode;
        *head = newNode;
    }
}

// Function to insert a node at the end of the circular doubly linked list
void insertAtLast(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        newNode->prev = newNode;
        newNode->next = newNode;
    } else {
        struct Node* last = (*head)->prev;
        newNode->next = *head;
        newNode->prev = last;
        (*head)->prev = newNode;
        last->next = newNode;
```

```
    }
}

// Function to delete the first node from the circular doubly linked list
void deleteAtFirst(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
    struct Node* temp = *head;
    if (temp->next == temp) {
        *head = NULL;
    } else {
        (*head)->prev->next = temp->next;
        temp->next->prev = (*head)->prev;
        *head = temp->next;
    }
    free(temp);
}

// Function to delete the last node from the circular doubly linked list
void deleteAtLast(struct Node** head) {
```

```
    if (*head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
    struct Node* temp = (*head)->prev;
    if (temp->next == temp) {
        *head = NULL;
    } else {
        temp->prev->next = *head;
        (*head)->prev = temp->prev;
    }
    free(temp);
}

// Function to traverse and print the circular doubly linked list
void traverse(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* current = head;
    do {
```

```
            printf("%d -> ", current->data);
            current = current->next;
        } while (current != head);
        printf("\n");
}

int main() {
        struct Node* head = NULL;

        // Insert elements at the beginning
        insertAtFirst(&head, 3);
        insertAtFirst(&head, 2);
        insertAtFirst(&head, 1);

        // Insert elements at the end
        insertAtLast(&head, 4);
        insertAtLast(&head, 5);

        // Print the linked list
        printf("Linked List:\n");
        traverse(head);
```

```
    // Delete elements from the beginning
    deleteAtFirst(&head);
    deleteAtFirst(&head);

    // Print the updated linked list
    printf("Linked List after deletion at the first:\n");
    traverse(head);

    // Delete elements from the end
    deleteAtLast(&head);
    deleteAtLast(&head);

    // Print the final linked list
    printf("Linked List after deletion at the last:\n");
    traverse(head);

    return 0;
}
```