Input code:

```
#include <stdio.h>

#include <stdlib.h>

// Define the structure for a node

struct Node {

    int data;

    struct Node* next;

    struct Node* prev;

};

// Function to create a new node

struct Node* createNode(int data) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = NULL;

    newNode->prev = NULL;

    return newNode;

}

// Function to insert a node at the beginning

void insertAtBeginning(struct Node** head, int data) {

    struct Node* newNode = createNode(data);

    if (*head == NULL) {

        *head = newNode;

    } else {

        newNode->next = *head;

        (*head)->prev = newNode;

        *head = newNode;

    }

}

// Function to insert a node at the end

void insertAtEnd(struct Node** head, int data) {

    struct Node* newNode = createNode(data);

    if (*head == NULL) {
        *head = newNode;

        return;

    }

    struct Node* temp = *head;

    while (temp->next != NULL) {

        temp = temp->next;

    }

    temp->next = newNode;

    newNode->prev = temp;

}

// Function to delete the first node

void deleteAtBeginning(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

void deleteAtEnd(struct Node** head) {

    if (*head == NULL) {

        printf("List is empty, cannot delete\n");

        return;

    }

    struct Node* temp = *head;

    while (temp->next != NULL) {

        temp = temp->next;

    }
    if (temp->prev != NULL) {
        temp->prev->next = NULL;
    } else {
        *head = NULL;
    }
    free(temp);
}
// Function to display the list from beginning to end
void displayForward(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
// Function to display the list from end to beginning
void displayBackward(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
```

```c
    }
    printf("\n");
}
// Main function to test the doubly linked list operations
int main() {
    struct Node* head = NULL;
    insertAtBeginning(&head, 10);
    insertAtEnd(&head, 20);
    insertAtEnd(&head, 30);
    insertAtBeginning(&head, 5);
    insertAtEnd(&head, 40);
    printf("Forward Traversal: ");
    displayForward(head);
    printf("Backward Traversal: ");
    displayBackward(head);
    deleteAtBeginning(&head);
    deleteAtEnd(&head);
    printf("After Deletion (Forward Traversal): ");
    displayForward(head);
    return 0;
}
```

Output code:

```
Forward Traversal: 5 10 20 30 40
Backward Traversal: 40 30 20 10 5
After Deletion (Forward Traversal): 10 20 30
```