Doubly Linked List in C

This document contains a C program demonstrating various operations on a Doubly Linked List. It includes insertion and deletion of nodes at the beginning, end, and a specific position, as well as functions for display, search, and exit.

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

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

struct Node* head = NULL;

// Create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->prev = newNode->next = NULL;
    return newNode;
}

// Insert at beginning
void insertAtBeginning(int value) {
    struct Node* newNode = createNode(value);
    if (head == NULL) {
        head = newNode;
        return;
    }
    newNode->next = head;
    head->prev = newNode;
    head = newNode;
}

// Insert at end
void insertAtEnd(int value) {
    struct Node* newNode = createNode(value);
    if (head == NULL) {
        head = newNode;
        return;
    }
    struct Node* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = newNode;
```

```
    newNode->prev = temp;
}

// Insert at specific position
void insertAtPosition(int value, int pos) {
    if (pos == 1) {
        insertAtBeginning(value);
        return;
    }
    struct Node* temp = head;
    for (int i = 1; i < pos - 1 && temp != NULL; i++) temp = temp->next;
    if (temp == NULL) return;
    struct Node* newNode = createNode(value);
    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next) temp->next->prev = newNode;
    temp->next = newNode;
}

// Delete from beginning
void deleteFromBeginning() {
    if (head == NULL) return;
    struct Node* temp = head;
    head = head->next;
    if (head) head->prev = NULL;
    free(temp);
}

// Delete from end
void deleteFromEnd() {
    if (head == NULL) return;
    struct Node* temp = head;
    while (temp->next != NULL) temp = temp->next;
    if (temp->prev) temp->prev->next = NULL;
    else head = NULL;
    free(temp);
}

// Delete from specific position
void deleteFromPosition(int pos) {
    if (head == NULL) return;
    if (pos == 1) {
        deleteFromBeginning();
        return;
    }
    struct Node* temp = head;
    for (int i = 1; i < pos && temp != NULL; i++) temp = temp->next;
    if (temp == NULL) return;
    if (temp->prev) temp->prev->next = temp->next;
    if (temp->next) temp->next->prev = temp->prev;
    free(temp);
}
```

```
// Search a node
void search(int value) {
    struct Node* temp = head;
    int pos = 1;
    while (temp != NULL) {
        if (temp->data == value) {
            printf("Value %d found at position %d\n", value, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Value %d not found\n", value);
}

// Display list
void display() {
    struct Node* temp = head;
    printf("List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    insertAtBeginning(10);
    insertAtEnd(20);
    insertAtPosition(15, 2);
    display();

    deleteFromBeginning();
    display();

    deleteFromEnd();
    display();

    insertAtEnd(30);
    insertAtEnd(40);
    insertAtEnd(50);
    display();

    deleteFromPosition(2);
    display();

    search(40);
    search(100);

    return 0;
}
```