Circular Doubly Linked List in C

A circular doubly linked list is a variation of a doubly linked list where the last node points to the first node and the first node points back to the last node. Each node contains three fields: two pointers (prev and next) and data.

Basic Operations in Circular Doubly Linked List:

1. Insertion at the beginning
2. Insertion at the end
3. Deletion from the beginning
4. Deletion from the end
5. Display the list

Example Code:

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

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

struct Node* head = NULL;
```

```
void insertAtEnd(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    if (head == NULL) {
        newNode->next = newNode->prev = newNode;
        head = newNode;
    } else {
        struct Node* tail = head->prev;
        newNode->next = head;
        head->prev = newNode;
        newNode->prev = tail;
        tail->next = newNode;
    }
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* tail = head->prev;
    if (tail == head) {
        free(head);
        head = NULL;
    } else {
        tail->prev->next = head;
```

```
        head->prev = tail->prev;
        free(tail);
    }
}

void display() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}

int main() {
    insertAtEnd(10);
    insertAtEnd(20);
    insertAtEnd(30);
    display();
    deleteFromEnd();
    display();
    return 0;
```

```
}
```