Doubly Circular Linked List

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

// Structure for a node in the doubly circular linked list
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to create a new node
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 insertAtBeginning(struct Node** head, int data) { struct
Node* newNode = createNode(data);
    if (*head == NULL) {
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        newNode->prev = (*head)->prev;
        newNode->next = *head;
        (*head)->prev->next = newNode;
        (*head)->prev = newNode;
    }
    *head = newNode;
}

// Function to insert a node at the end of the linked list
void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
```

```
        newNode->next = newNode;
        newNode->prev = newNode;
        *head = newNode;
    } else {
        newNode->prev = (*head)->prev;
        newNode->next = *head;
        (*head)->prev->next = newNode;
        (*head)->prev = newNode;
    }
}

// Function to insert a node at a given position in the linked list
void insertAtPosition(struct Node** head, int data, int position) {
if (position <= 0) {
        insertAtBeginning(head, data);
        return;
    }

    struct Node* newNode = createNode(data);
    struct Node* current = *head;

    for (int i = 1; i < position && current->next != *head; ++i) {
        current = current->next;
    }

    newNode->next = current->next;
    newNode->prev = current;
    current->next->prev = newNode;
    current->next = newNode;
}

// Function to delete the first node of the linked list
void deleteAtBeginning(struct Node** head) {
    if (*head == NULL)
        return;

    struct Node* temp = *head;

    if ((*head)->next == *head) {
        *head = NULL;
    } else {
        (*head)->next->prev = (*head)->prev;
        (*head)->prev->next = (*head)->next;
```

```
        *head = (*head)->next;
    }

    free(temp);
}

// Function to delete the last node of the linked list
void deleteAtEnd(struct Node** head) {
    if (*head == NULL)
        return;

    struct Node* temp = (*head)->prev;

    if ((*head)->next == *head) {
        *head = NULL;
    } else {
        (*head)->prev = temp->prev;
        temp->prev->next = *head;
    }

    free(temp);
}

// Function to delete a node at a given position in the linked list
void deleteAtPosition(struct Node** head, int position) { if
(*head == NULL)
        return;

    if (position <= 0) {
        deleteAtBeginning(head);
        return;
    }

    struct Node* current = *head;

    for (int i = 1; i < position && current->next != *head; ++i) {
        current = current->next;
    }

    if (current->next == *head) {
        return; // Node not found at given position
    }
```

```
    struct Node* temp = current->next;

    if (temp->next == *head) {
        current->next = *head;
        (*head)->prev = current;
    } else {
        current->next = temp->next;
        temp->next->prev = current;
    }

    free(temp);
}

// Function to search for a node with a given data value
struct Node* search(struct Node* head, int target) { if
(head == NULL)
        return NULL;

    struct Node* current = head;
    do {
        if (current->data == target) {
            return current;
        }
        current = current->next;
    } while (current != head);

    return NULL;
}

// Function to display the linked list
void display(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 at beginning
    insertAtBeginning(&head, 10);
    insertAtBeginning(&head, 5);
    printf("After inserting at beginning: ");
    display(head);
    // Insert at end
    insertAtEnd(&head, 20);
    insertAtEnd(&head, 25);
    printf("After inserting at end: ");
    display(head);
    // Insert at position
    insertAtPosition(&head, 15, 2);
    printf("After inserting at position 2: ");
    display(head);
    // Delete at beginning
    deleteAtBeginning(&head);
    printf("After deleting at beginning: ");
    display(head);

    // Delete at end
    deleteAtEnd(&head);
    printf("After deleting at end: ");
    display(head);

    // Delete at position
    deleteAtPosition(&head, 2);
    printf("After deleting at position 2: ");
    display(head);
    // Search
    int searchValue = 15;
    struct Node* searchResult = search(head, searchValue);
    if (searchResult != NULL) {
        printf("Found %d in the list.\n", searchValue);
    } else {
        printf("%d not found in the list.\n",
    searchValue); }
        return 0;
}
```