Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology

Name :Dumpeta Varshitha . Email :varshithadumpeta@gmail.com
Roll no:23071A1023 Phone :8919552327
Branch :VNRVJIET Department :EIE
Batch :2027 Degree :B.Tech EIE

2023_27_II_EIE 1_Data Structures_lab

DATA STRUCTURES_CODING_WEEK 6

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : CODING

1. Problem Statement

Write a program to perform and implement a Circular singly-linked list. It should be a menu-driven program with the following functions:

insert() - Inserts node at the beginningbeginDelete() - Deletes node from the beginninglastDelete() - Deletes node from the endrandomDelete() - Deletes node with specific datasearch() - Search for a node with a specific nodedisplay() - Displays the listexit - Exits the program

Answer

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

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

struct CircularLinkedList {
```

```
    struct Node* head;
};

void insert(struct CircularLinkedList* list, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (list->head == NULL) {
        list->head = newNode;
        newNode->next = list->head;
    } else {
        struct Node* curr = list->head;
        while (curr->next != list->head) {
            curr = curr->next;
        }
        curr->next = newNode;
        newNode->next = list->head;
        list->head = newNode;
    }
    printf("Node with data %d inserted.\n", data);
}

void beginDelete(struct CircularLinkedList* list) {
    if (list->head == NULL) {
        printf("Circular Linked List is empty.\n");
        return;
    }

    if (list->head->next == list->head) {
        list->head = NULL;
    } else {
        struct Node* curr = list->head;
        while (curr->next != list->head) {
            curr = curr->next;
        }
        curr->next = list->head->next;
        list->head = list->head->next;
    }

    printf("Node at the beginning deleted.\n");
}
```

```
void lastDelete(struct CircularLinkedList* list) {
    if (list->head == NULL) {
        printf("Circular Linked List is empty.\n");
        return;
    }

    if (list->head->next == list->head) {
        list->head = NULL;
    } else {
        struct Node* curr = list->head;
        struct Node* prev = NULL;
        while (curr->next != list->head) {
            prev = curr;
            curr = curr->next;
        }
        prev->next = list->head;
    }

    printf("Node at the end deleted.\n");
}

void randomDelete(struct CircularLinkedList* list, int data) {
    if (list->head == NULL) {
        printf("Circular Linked List is empty.\n");
        return;
    }

    if (list->head->data == data) {
        beginDelete(list);
        return;
    }

    struct Node* curr = list->head;
    struct Node* prev = NULL;
    int found = 0;
    do {
        if (curr->data == data) {
            found = 1;
            break;
        }
        prev = curr;
```

```
        curr = curr->next;
    } while (curr != list->head);

    if (!found) {
        printf("Node with data %d not found.\n", data);
        return;
    }

    prev->next = curr->next;
    printf("Node with data %d deleted.\n", data);
}

void search(struct CircularLinkedList* list, int data) {
    if (list->head == NULL) {
        printf("Circular Linked List is empty.\n");
        return;
    }

    struct Node* curr = list->head;
    int found = 0;
    int position = 1;
    do {
        if (curr->data == data) {
            found = 1;
            break;
        }
        curr = curr->next;
        position++;
    } while (curr != list->head);

    if (found) {
        printf("Node with data %d found at position %d.\n", data, position);
    } else {
        printf("Node with data %d not found.\n", data);
    }
}

void display(struct CircularLinkedList* list) {
    if (list->head == NULL) {
        printf("Circular Linked List is empty.\n");
        return;
    }
```

```
    printf("Circular Linked List: ");
    struct Node* curr = list->head;
    do {
        printf("%d ", curr->data);
        curr = curr->next;
    } while (curr != list->head);
    printf("\n");
}

int main() {
    struct CircularLinkedList circularLinkedList;
    circularLinkedList.head = NULL;
    int choice, data;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                scanf("%d", &data);
                insert(&circularLinkedList, data);
                break;
            case 2:
                beginDelete(&circularLinkedList);
                break;
            case 3:
                lastDelete(&circularLinkedList);
                break;
            case 4:
                scanf("%d", &data);
                randomDelete(&circularLinkedList, data);
                break;
            case 5:
                scanf("%d", &data);
                search(&circularLinkedList, data);
                break;
            case 6:
                display(&circularLinkedList);
                break;
            case 7:
                exit(0);
```

```
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

2. Problem Statement

You are given the task of implementing a program that works with a doubly linked list. The program should support the following operations:

Add Node: Allow the user to input a number of nodes and their respective data to construct a doubly linked list.Display List: Display the elements of the doubly linked list in the order they were added.

Write a program to accomplish the above tasks.

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

struct DoublyLinkedList {
    struct Node* head;
    struct Node* tail;
    int size;
};

void initDoublyLinkedList(struct DoublyLinkedList* list) {
    list->head = NULL;
```

```
    list->tail = NULL;
    list->size = 0;
}

void addNode(struct DoublyLinkedList* list, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->previous = NULL;
    newNode->next = NULL;

    if (list->size == 0) {
        list->head = list->tail = newNode;
    } else {
        newNode->previous = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }

    list->size++;
}

void display(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    if (list->size == 0) {
        printf("List is empty\n");
        return;
    }

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

int main() {
    struct DoublyLinkedList my_list;
    initDoublyLinkedList(&my_list);

    int n;
    scanf("%d", &n);
```

```
    if (n == 0) {
        printf("List is empty\n");
    } else {
        int element;
        for (int i = 0; i < n; i++) {
            scanf("%d", &element);
            addNode(&my_list, element);
        }

        display(&my_list);
    }

    // Free memory for the linked list nodes
    struct Node* current = my_list.head;
    while (current != NULL) {
        struct Node* nextNode = current->next;
        free(current);
        current = nextNode;
    }
    return 0;
}
```

Status : Correct ***Marks : 10/10***