Practical no:- 07

Implementation of Circular linked list.

AIM:- Implement a Circular Single Linked List (CSLL) and perform the operations: Create, Traverse, Insert_Beg, Insert_End, Delete_beg, Delete_end using Menu Driver Program.

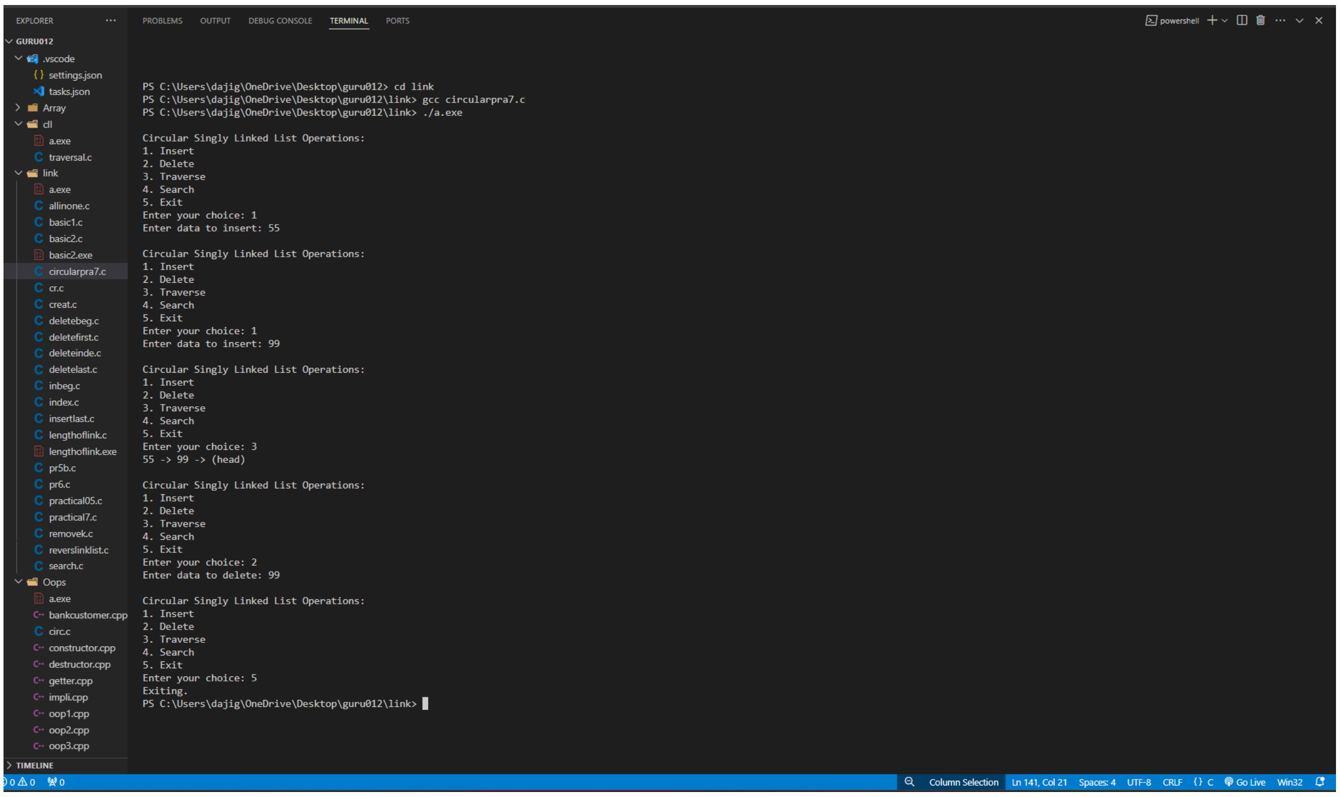
Program:-

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

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

typedef struct CircularLinkedList {
   Node* head;
} CircularLinkedList;

CircularLinkedList* createList() {
   CircularLinkedList* cll = (CircularLinkedList*)malloc(sizeof(CircularLinkedList));
   cll->head = NULL;
   return cll;
}

void insert(CircularLinkedList* cll, int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;

   if (cll->head == NULL) {
      cll->head = newNode;
      newNode->next = cll->head;
```

```
    } else {
        Node* temp = cll->head;
        while (temp->next != cll->head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = cll->head;
    }
}

void deleteNode(CircularLinkedList* cll, int key) {
    if (cll->head == NULL) return;

    Node* current = cll->head;
    Node* prev = NULL;

    // If the node to be deleted is the head node
    if (current->data == key) {
        if (current->next == cll->head) {
            free(current);
            cll->head = NULL;
        } else {
            while (current->next != cll->head) {
                current = current->next;
            }
            current->next = cll->head->next;
            Node* temp = cll->head;
            cll->head = cll->head->next;
            free(temp);
        }
        return;
    }

    // Traverse the list to find the node to delete
    while (current->next != cll->head && current->next->data != key) {
        current = current->next;
    }

    // If node with key is found
    if (current->next->data == key) {
        Node* temp = current->next;
        current->next = temp->next;
        free(temp);
    }
}

void traverse(CircularLinkedList* cll) {
    if (cll->head == NULL) {
        printf("List is empty.\n");
        return;
    }

    Node* temp = cll->head;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != cll->head);
    printf("(head)\n");
}

int search(CircularLinkedList* cll, int key) {
    if (cll->head == NULL) return 0;
    Node* temp = cll->head;
    do {
        if (temp->data == key) {
            return 1; // found
        }
        temp = temp->next;
    } while (temp != cll->head);

    return 0; // not found
}

void menu() {
    CircularLinkedList* cll = createList();
    int choice, data;

    while (1) {
        printf("\nCircular Singly Linked List Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Traverse\n");
        printf("4. Search\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insert(cll, data);
                break;
            case 2:
                printf("Enter data to delete: ");
                scanf("%d", &data);
                deleteNode(cll, data);
                break;
            case 3:
                traverse(cll);
                break;
            case 4:
                printf("Enter data to search: ");
                scanf("%d", &data);
                if (search(cll, data)) {
                    printf("%d found in the list.\n", data);
                } else {
                    printf("%d not found in the list.\n", data);
                }
                break;
            case 5:
                printf("Exiting.\n");
                free(cll);
                return;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}

int main() {
    menu();
    return 0;
```

```
}
```

[OUTPUT]



```
PS C:\Users\dajig\OneDrive\Desktop\guru012> cd link
PS C:\Users\dajig\OneDrive\Desktop\guru012\link> gcc circularpra7.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\link> ./a.exe

Circular Singly Linked List Operations:
1. Insert
2. Delete
3. Traverse
4. Search
5. Exit
Enter your choice: 1
Enter data to insert: 55

Circular Singly Linked List Operations:
1. Insert
2. Delete
3. Traverse
4. Search
5. Exit
Enter your choice: 1
Enter data to insert: 99

Circular Singly Linked List Operations:
1. Insert
2. Delete
3. Traverse
4. Search
5. Exit
Enter your choice: 3
55 -> 99 -> (head)

Circular Singly Linked List Operations:
1. Insert
2. Delete
3. Traverse
4. Search
5. Exit
Enter your choice: 2
Enter data to delete: 99

Circular Singly Linked List Operations:
1. Insert
2. Delete
3. Traverse
4. Search
5. Exit
Enter your choice: 5
Exiting.
PS C:\Users\dajig\OneDrive\Desktop\guru012\link> 
```

Github Link:- https://github.com/guru24961/Data-Stracture-practical.git