



**Vidyavardhini's College of Engineering and Technology**

**Department of Artificial Intelligence & Data Science**

<b>Experiment No.7</b>	
Implement Circular Linked List ADT.	
Name: Gautam D. Chaudhari	
Roll No: 04	
Date of Performance:	
Date of Submission:	
Marks:	
Sign:	



### Experiment No. 7: Circular Linked List Operations

#### Aim: Implementation of Circular Linked List ADT

#### Objective:

In circular linked list last node is connected to first node. On other hand circular linked list can be used to implement traversal along web pages.

#### Theory:

In a circular linked list, the last node contains a pointer to the first node of the list. We can have a circular singly linked list as well as a circular doubly linked list. While traversing a circular linked list, we can begin at any node and traverse the list in any one direction, forward or backward, until we reach the same node where we started. Thus, a circular linked list has no beginning and no ending.

#### Inserting a New Node in a Circular Linked List

Case 1: The new node is inserted at the beginning.

Case 2: The new node is inserted at the end.

#### Deleting a Node from a Circular Linked List

Case 1: The first node is deleted.

Case 2: The last node is deleted.

Insertion and Deletion after or before a given node is same as singly linked list.

#### Algorithm

Algorithm to insert a new node at the beginning

Step 1: IF AVAIL = NULL

    Write OVERFLOW

    Go to Step 9 [END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL → NEXT

Step 4: SET NEW\_NODE-->DATA = VAL

Step 5: SET PTR=START

Repeat Step 6 while PTR NEXT != START

Step 6: SET PTR = PTR NEXT [END OF LOOP]



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

Step 7: SET NEW\_NODE--> NEXT= START

Step 8: SET PTR-->NEXT = START

Step 9: SET START = NEW\_NODE

Step 10: EXIT

Algorithm to insert a new node at the end

Step 1: IF AVAIL = NULL

    Write OVERFLOW

    Go to Step 11 [END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL--> NEXT

Step 4: SET NEW\_NODE -->DATA = VAL

Step 5: SET NEW\_NODE-->NEXT = START

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR--> NEXT != START

Step 8: SET PTR = PTR -->NEXT [END OF LOOP]

Step 9: SET PTR -->NEXT = NEW\_NODE

Step 10: EXIT

Algorithm to delete the first node

Step 1: IF START = NULL

    Write UNDERFLOW

    Go to Step 6 [END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 while PTR--> NEXT != START

Step 4: SET PTR = PTR -->NEXT [END OF LOOP]

Step 4: SET PTR →NEXT = START -->NEXT

Step 5: FREE START

Step 6: EXIT

Algorithm to delete the last node

Step 1: IF START = NULL

    Write UNDERFLOW



Go to Step 7 [END OF IF]

Step 2: SET PTR = START [END OF LOOP]

Step 3: Repeat Step 4 and Step 5 while PTR -->NEXT != START

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR -->NEXT

Step 6: SET PREPTR-->NEXT = START

Step 7: FREE PTR

Step 8: EXIT

### Code:

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
    int data;
    struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *);
struct node *display(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_after(struct node *);
struct node *delete_list(struct node *);
int main()
{
    int option;
    clrscr();
    do
    {
        printf("\n\n **MAIN MENU **");
        printf("\n 1: Create a list");
        printf("\n 2: Display the list");
        printf("\n 3: Add a node at the beginning");
        printf("\n 4: Add a node at the end");
        printf("\n 5: Delete a node from the beginning");
        printf("\n 6: Delete a node from the end");
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
printf("\n 7: Delete a node after a given node");
printf("\n 8: Delete the entire list");
printf("\n 9: EXIT");
printf("\n\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
    case 1: start = create_cll(start);
    printf("\n CIRCULAR LINKED LIST CREATED");
    break;
    case 2: start = display(start);
    break;
    case 3: start = insert_beg(start);
    break;
    case 4: start = insert_end(start);
    break;
    case 5: start = delete_beg(start);
    break;
    case 6: start = delete_end(start);
    break;
    case 7: start = delete_after(start);
    break;
    case 8: start = delete_list(start);
    printf("\n CIRCULAR LINKED LIST DELETED");
    break;
}
}while(option !=9);
getch();
return 0;
}
struct node *create_cll(struct node *start)
{
    struct node *new_node, *ptr;
    int num;
    printf("\n Enter -1 to end");
    printf("\n Enter the data : ");
    scanf("%d", &num);
    while(num!= -1)
    {
        new_node = (struct node*)malloc(sizeof(struct node));
        new_node -> data = num;
        if(start == NULL)
        {
```



```
new_node -> next = new_node;
start = new_node;
}
else
{ ptr = start;
while(ptr -> next != start)
ptr = ptr -> next;
ptr -> next = new_node;
new_node -> next = start;
}
printf("\n Enter the data : ");
scanf("%d", &num);
}
return start;
}
struct node *display(struct node *start)
{
struct node *ptr;
ptr=start;
while(ptr -> next != start)
{
printf("\t %d", ptr -> data);
ptr = ptr -> next;
}
printf("\t %d", ptr -> data);
return start;
}
struct node *insert_beg(struct node *start)
{
struct node *new_node, *ptr;
int num;
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
ptr = start;
while(ptr -> next != start)
ptr = ptr -> next;
ptr -> next = new_node;
new_node -> next = start;
start = new_node;
return start;
}
```



**Vidyavardhini's College of Engineering and Technology**

**Department of Artificial Intelligence & Data Science**

```
struct node *insert_end(struct node *start)
{
    struct node *ptr, *new_node;
    int num;
    printf("\n Enter the data : ");
    scanf("%d", &num);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node->data = num;
    ptr = start;
    while(ptr->next != start)
        ptr = ptr->next;
    ptr->next = new_node;
    new_node->next = start;
    return start;
}
struct node *delete_beg(struct node *start)
{
    struct node *ptr;
    ptr = start;
    while(ptr->next != start)
        ptr = ptr->next;
    ptr->next = start->next;
    free(start);
    start = ptr->next;
    return start;
}
struct node *delete_end(struct node *start)
{
    struct node *ptr, *preptr;
    ptr = start;
    while(ptr->next != start)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = ptr->next;
    free(ptr);
    return start;
}
struct node *delete_after(struct node *start)
{
    struct node *ptr, *preptr;
    int val;
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
printf("\n Enter the value after which the node has to deleted : ");
scanf("%d", &val);
ptr = start;
preptr = ptr;
while(preptr -> data != val)
{
    preptr = ptr;
    ptr = ptr -> next;
}
preptr -> next = ptr -> next;
if(ptr == start)
    start = preptr -> next;
free(ptr);
return start;
}
struct node *delete_list(struct node *start)
{
    struct node *ptr;
    ptr = start;
    while(ptr -> next != start)
        start = delete_end(start);
    free(start);
    return start;
}
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

#### Output:

```
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Delete the entire list
9: EXIT

Enter your option : 1
Enter -1 to end.
Enter data: 20 80 4 5 3 -1
Enter data: Enter data: Enter data: Enter data: Enter data: CIRCULAR LINKED LIST
CREATED
```

```
**MAIN MENU ***
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Delete the entire list
9: EXIT
```

```
Enter your option :
```

```
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Delete the entire list
9: EXIT
```

```
Enter your option : 2
```

```
20 80 4 5 3
```

```
**MAIN MENU ***
1: Create a list
2: Display the list
3: Add a node at the beginning
4: Add a node at the end
5: Delete a node from the beginning
6: Delete a node from the end
7: Delete a node after a given node
8: Delete the entire list
9: EXIT
```

```
Enter your option :
```



#### Conclusion:

Write an example of insertion and deletion in the circular linked list while traversing the web pages?

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

// Structure to represent a web page
struct WebPage {
    char url[100];
    struct WebPage* next;
};

// Global pointer to the current web page
struct WebPage* currentWebPage = NULL;

// Function to insert a new web page after the current web page
void insertWebPage(struct WebPage* currentPage, const char* url) {
    struct WebPage* newPassword = (struct WebPage*)malloc(sizeof(struct WebPage));
    strcpy(newPage->url, url);

    if (currentWebPage == NULL) {
        // If the list is empty, make the new page the current page and point to itself
        currentWebPage = newPassword;
        currentWebPage->next = currentWebPage;
    } else {
        // Insert the new page after the current page
        newPassword->next = currentPage->next;
        currentPage->next = newPassword;
    }
}
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
currentWebPage = currentPage;  
}  
}
```

```
// Function to delete the current web page
```

```
void deleteWebPage() {  
    if (currentPage == NULL) {  
        printf("No web page to delete.\n");  
        return;  
    }
```

```
    struct WebPage* pageToDelete = currentPage;
```

```
    struct WebPage* prevPage = currentPage;
```

```
// Find the page before the current page
```

```
    while (prevPage->next != currentPage) {  
        prevPage = prevPage->next;  
    }
```

```
// Update the previous page to skip the current page
```

```
    prevPage->next = currentPage->next;
```

```
// If deleting the only page, set current page to NULL
```

```
    if (prevPage == currentPage) {  
        currentPage = NULL;  
    } else {  
        currentPage = prevPage;  
    }
```

```
    free(pageToDelete);
```

```
}
```

```
// Function to display the list of web pages
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
void displayWebPages() {  
    if (currentWebPage == NULL) {  
        printf("No web pages to display.\n");  
        return;  
    }  
  
    struct WebPage* currentPage = currentWebPage;  
  
    do {  
        printf("Web Page: %s\n", currentPage->url);  
        currentPage = currentPage->next;  
    } while (currentPage != currentWebPage);  
}  
  
int main() {  
    int option;  
    char url[100];  
  
    do {  
        printf("\n\n **Web Browser Simulation **");  
        printf("\n 1: Insert a new web page");  
        printf("\n 2: Delete the current web page");  
        printf("\n 3: Display web pages");  
        printf("\n 4: EXIT");  
        printf("\n\n Enter your option: ");  
        scanf("%d", &option);  
  
        switch (option) {  
            case 1:  
                printf("Enter the URL of the new web page: ");  
                scanf("%s", url);  
                insertWebPage(currentWebPage, url);  
                break;  
        }  
    } while (option != 4);  
}
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

case 2:

```
    deleteWebPage();  
    break;
```

case 3:

```
    displayWebPages();  
    break;
```

```
}
```

```
} while (option != 4);
```

```
return 0;
```

```
}
```