Experiment No.7
Implement Circular Linked List ADT.
Name: Shravani Sandeep Raut
Roll No: 48
Date of Performance:
Date of Submission:

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]

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<malloc.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;

void add_at_begin()
{
    struct node *temp;
    struct node *p;
    int num;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("Enter the number to perform add at begin:");
    scanf("%d",&num);
    if(head==NULL)
    {
       temp->data=num;
       temp->next=temp;
       head=temp;
    }
    else
    {
       temp->data=num;
       temp->next=head;
       p=head;
       while(p->next!=head)
       {
          p=p->next;
       }
       p->next=temp;
       head=temp;
   }
```

```
}

void add_at_end()
{
    struct node *temp,*p;
    int num;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter the number to perform add at END:");
    scanf("%d",&num);
    temp->data=num;
    temp->next=head;
    if(head==NULL)
    {
       temp->data=num;
       temp->next=temp;
       head=temp;
    }
    else
    {
       p=head;
       while(p->next!=head)
       {
         p=p->next;
       }
       p->next=temp;
      temp->next=head;
   }
}

void add_in_between()
{
    int pos, num,i;
    struct node*temp,*p;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter the number to perform add in between:");
    scanf("%d",&num);
    temp->data=num;
    printf("Enter the position");
    scanf("%d",&pos);
    p=head;
    for(i=1;i<=pos-1;i++)
    {
       p=p->next;
    }
    temp->next=p->next;
    p->next=temp;
}

void search()
```

```
{
    struct node*p=head;
    int r=0;
    int num;
    printf("\nEnter the number to be search:");
    scanf("\n%d",&num);
    do
    {
        if(p->data==num)
        {
           r=1;
           break;
        }
    p=p->next;
    } while(p!=head);

    if(r==0)
    {
      printf("Number is not present");
    }
    else
    {
       printf("Number is present");
    }
}

void delete_begin()
{
   struct node *p,*q;
   p=head;
   q=head;
   while(p->next!=head)
   {
      p=p->next;
   }
   head=head->next;
   p->next=head;
   q->next=NULL;
}

void delete_end()
{
    struct node*p,*q;
    p=head;
    while(p->next!=head)
    {
       q=p;
       p=p->next;
    }
```

```
    q->next=head;
    p->next=NULL;

}

void delete_in_between()
{
    struct node *p, *q;
    int pos,i;
    p=head;
    printf("\nEnter the position from which you want to delete number:");
    scanf("\n%d",&pos);
    for(i=1;i<=pos;i++)
    {
       q=p;
       p=p->next;
    }
    q->next=p->next;
    p->next=NULL;

}

void display()
{
    struct node *p;
    p=head;
    printf("Element of linked list:");
    while(p->next!=head)
    {
       printf("\n%d",p->data);
       p=p->next;
    }
   printf("\n%d",p->data);

}

void main()
{
    add_at_begin();
    add_at_begin();
    add_at_begin();
    add_at_begin();
    display();
    add_at_end();
    display();
    add_in_between();
    display();
    search();
    delete_in_between(head);
```

```
    display(head);
    delete_begin();
    display();
    delete_end();
    display();
}
```

Output:

```
Enter the number to perform add at begin:10
Enter the number to perform add at begin:20
Enter the number to perform add at begin:30
Enter the number to perform add at begin:40
Element of linked list:
40
30
20
10
Enter the number to perform add at END:100
Element of linked list:
40
30
20
10
100
Enter the number to perform add in between:50
Enter the position2
Element of linked list:
40
30
50
20
10
100
Enter the number to be search:30
Number is present
Enter the position from which you want to delete number:3
```

Conclusion:

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

Traversal

- **Purpose**: Manage the history of visited web pages in a web browser.

Insertion

1. **Start**: User visits a new page (e.g., `page1.com`).
2. **Create Node**: A new node is created for the URL.
3. **Check List**:
    - If the list is empty, set the new node as the head and point it to itself.
    - If not, traverse to the last node and link it to the new node, then point the new node to the head (circular link).

Deletion

1. **Identify Node**: User wants to remove a specific page (e.g., `page2.com`).
2. **Check Head**:
    - If the head matches the URL, update the head to the next node.
    - If there's only one node, set the list to empty.
3. **Traverse**: If not at the head, traverse the list to find the node.
4. **Bypass Node**: Adjust pointers to bypass the node to be deleted.

Traversal

1. **Start from Head**: Begin at the head of the list.
2. **Print URLs**: Loop through the list, printing each URL until reaching the head again, ensuring continuous traversal due to the circular nature.

Conclusion

Using a circular linked list efficiently manages web page history, allowing for dynamic insertion and deletion, while enabling seamless navigation through the list of visited pages.