Syeda Reeha Quasar

14114902719

3C7

Aim

Create a circular linked list and perform following operations: Insertion at front and Deletion at end.

Experiment - 7

Data Structures

# **EXPERIMENT – 7**

**AIM:** Create a circular linked list and perform following operations: Insertion at front and Deletion at end.

# **THEORY**

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.



**Advantages of Circular Linked Lists:**

1. Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.
2. Useful for implementation of queue. Unlike this implementation, we don’t need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.
3. Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.
4. Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.

## **INSERTION AT THE BEGINNING CLL**

## **Source code:**

// insertion in the beginning of circular linked list

//required libraries

#include <stdio.h>

#include <stdlib.h>

//node declaration

struct node

{

int data;

struct node \*next;

}

\*head=NULL,\*last=NULL; // initialising

void createCLL(int givenData)

{

struct node \*newNode; // creating a node to be added

newNode = (struct node\*)malloc(sizeof(struct node)); // allocating memory

//storing value in new node and pointing it to null

newNode -> data = givenData;

newNode -> next = NULL;

if(head==NULL) // checking if head is null

{

head = last = newNode; //assigning head and last as new node in case of empty list

}

else

{

last -> next = newNode; // attaching new element to last

last = last -> next; // updating last

}

}

void InsertAtBeginning(int givenData)

{

struct node\* current = head; // current node points to head using for traversal

struct node\* newNode = (struct node\*)malloc(sizeof(struct node)); // creating and allocating memory to newNode

newNode -> data = givenData; // storing given value to a newNode

newNode -> next = head; // pointing newNode to head

while (current -> next != head) // finding last element

{

current = current -> next;

}

current -> next = newNode; //making node last element so we can remove head and attach this instead

head = newNode; // updating head

}

void display(struct node \*cll)

{

// traversing and printinglist elements

int i=1;

printf("\n Circular linked list formed is:\n");

do {

printf("\nNode Present at position %d = %d\n", i, cll -> data);

cll = cll -> next;

i++;

}while (cll!=head); // using do while loop to print head then check condition

}

int main()

{

// my info

printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group - 3C7 \n\n");

int n, x, value, i;

printf("\nEnter No. of Nodes to be inserted: ");

scanf("%d",&n);

for (i = 1; i <= n; i++)

{

printf("\nEnter Data of Node at position %d :",i); // taking data for nodes

scanf("%d",&x);

createCLL(x);

// condition to make linked list circular by connecting last and head

if (i == n) {

last -> next = head;

}

}

printf("\n Circular Linked list Before Insertion\n"); // printing cll before insertion

display(head);

printf("\nEnter Data of Node to be inserted at front:"); //taking input necessary and calling func

scanf("%d",&value);

InsertAtBeginning(value);

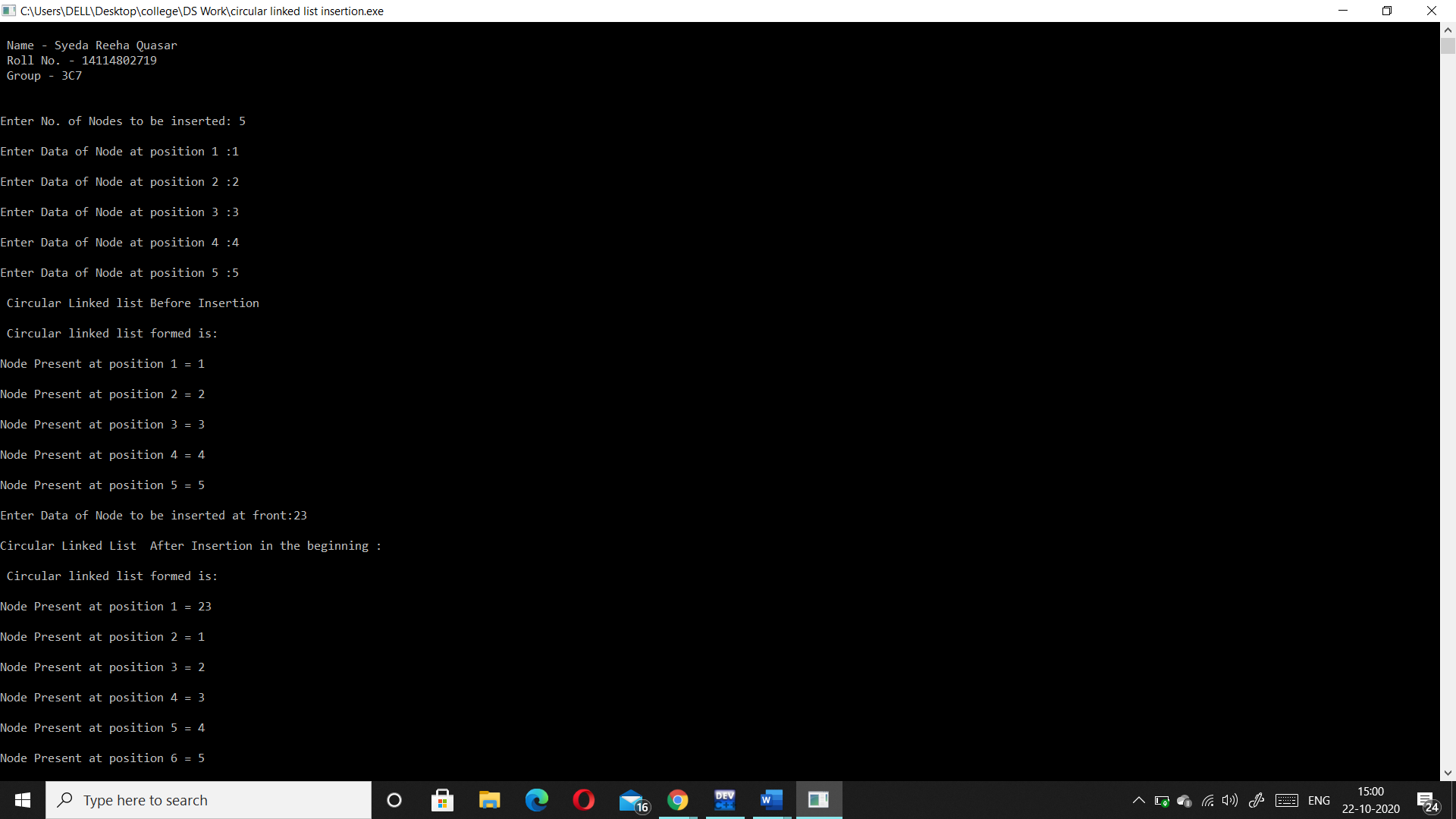
//printing after insertion

printf("\nCircular Linked List After Insertion in the beginning :\n");

display(head);

return 0;

}

**OUTPUT** 

**DELETION FROM THE END CLL**

**Source Code:**

// insertion in the beginning of circular linked list

//required libraries

#include <stdio.h>

#include <stdlib.h>

//node declaration

struct node

{

int data;

struct node \*next;

}

\*head=NULL,\*last=NULL; // initialising

void createCLL(int givenData)

{

struct node \*newNode; // creating a node to be added

newNode = (struct node\*)malloc(sizeof(struct node)); // allocating memory

//storing value in new node and pointing it to null

newNode -> data = givenData;

newNode -> next = NULL;

if(head==NULL) // checking if head is null

{

head = last = newNode; //assigning head and last as new node in case of empty list

}

else

{

last -> next = newNode; // attaching new element to last

last = last -> next; // updating last

}

}

void DeletionAtEnd()

{

struct node\* secondlast = head; // creating a pointer for head

while (secondlast -> next -> next != head) // finding second last element

{

secondlast = secondlast -> next;

}

secondlast -> next = head; //updating head

}

void display(struct node \*cll)

{

// traversing and printinglist elements

int i=1;

do {

printf(" %d ", cll -> data);

cll = cll -> next;

i++;

}while (cll!=head); // using do while loop to print head then check condition

}

int main()

{

// my info

printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group - 3C7 \n\n");

int n, x, value, i;

printf("\nEnter No. of Nodes to be inserted: ");

scanf("%d",&n);

for (i = 1; i <= n; i++)

{

printf("\nEnter Data of Node at position %d :",i); // taking data for nodes

scanf("%d",&x);

createCLL(x);

// condition to make linked list circular by connecting last and head

if (i == n) {

last -> next = head;

}

}

printf("\n Circular Linked list Before deletion\n"); // printing cll before insertion

display(head);

printf("\n----------- Deleting element from the last ----------\n"); // deleting element from the last

DeletionAtEnd(value);

//printing after insertion

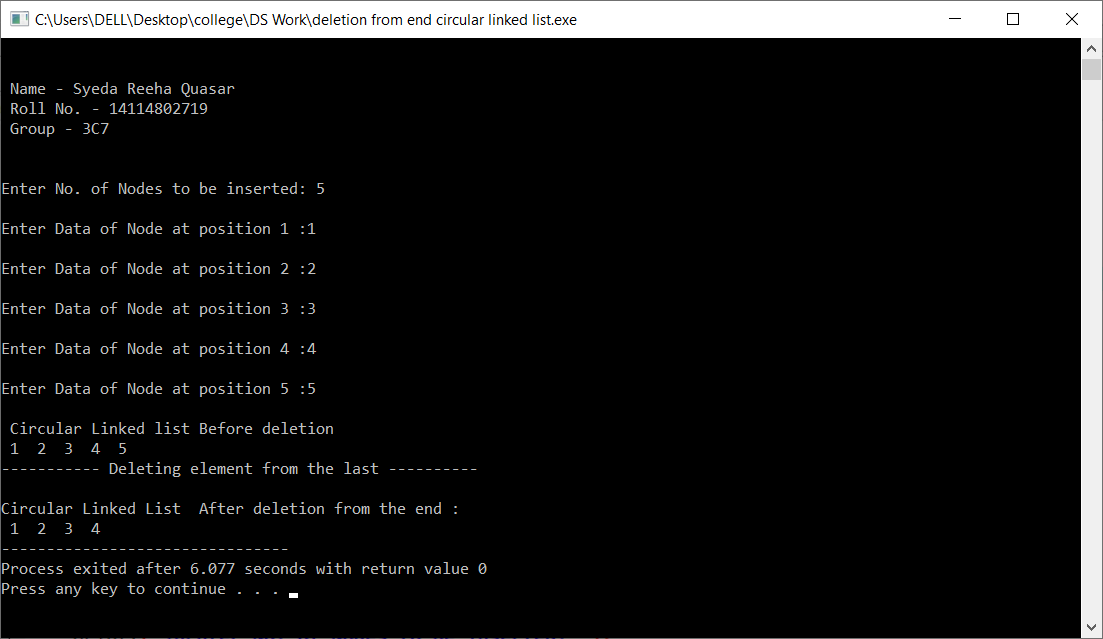
printf("\nCircular Linked List After deletion from the end :\n");

display(head);

return 0;

}

**OUTPUT**



# **VIVA VOICE**

**Q1. Is there any Null pointer in circular linked list?**

Ans.

NO, there is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list. ... We can maintain a pointer to the last inserted node and front can always be obtained as next of last.

**Q2. What is circular linked list?**

Ans.

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

**Q3. Whether a circular linked list is single way list or two way list?**

Ans.

A circular linked list can be single way or two way depending on whether it is a singly circular linked list or doubly circular linked list. We can travel two way i.e. clockwise and anti- clockwise in a doubly linked list while in singly linked list we can only travel in clockwise direction.

**Q4. Which pointer signifies the starting of circular linked list?**

Ans.

Head is the starting of circular linked list. The first Node is the Head for any Linked List. When a new Linked List is instantiated, it just has the Head, which is Null. Else, the Head holds the pointer to the fisrt Node of the List.

**Q5. Describe the steps to delete the starting node of linked list.**

Ans.

1. Find the previous node of the node to be deleted.
2. Change the next of the previous node.
3. Free memory for the node to be deleted.